#131 SEPTEMBER 1987

DR. DOBB'S JOURNAL

2.95 (3.95 CANADA)

Dr. Dobb's Journal of Software Tools PROFESSIONAL PROGRAMMER

Quest for Algorithms

SEGIM:

OK-OOT-L

Writing MS-DOS Device Drivers

Languages: C, Pascal, Smalltalk, and V.I.P.



131 SEPTEMBER 1987

oo C: **NEW!** erful optimizing ler ever

Sieve benchmark

	Turbo C	Microsoft* C
Compile time	2.4	13.51
Compile and link time	4.1	18.13
Execution time	3.95	5.93
Object code size	239	249
Execution size	5748	7136
Price	\$99.9 5	\$450.00

Benchmark run on an IBM PS/2 Model 60 using Turbo C version 1.0 and the Turbo Linker version 1.0; Microsoft C version 4.0 and the MS overlay linker version 3.51.

Technical Specifications

- Compiler: One-pass optimizing compiler generating linkable object modules. Included is Borland's high-performance Turbo Linker." The object module is compatible with the PC-DOS linker. Supports tiny, small, compact, medium, large, and huge memory model libraries. Can mix models with near and far pointers. Includes floating point emulator (utilizes 8087/80287 if installed).
- ☑ Interactive Editor: The system includes a powerful, interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code.
- Development Environment: A powerful "Make" is included so that managing Turbo C program development is highly efficient. Also includes pulldown menus and windows.
- Links with relocatable object modules created using Borland's Turbo Prolog into a single program.
- ☑ Inline assembly code.
- ☑ Loop optimizations.
- Register variables.
- ANSI C compatible.
- Start-up routine source code included.
- Both command line and integrated environment versions included.
- License to the source code for Runtime Library available.

Join more than 100,000 Turbo C enthusiasts. Get your copy of Turbo C today!

Minimum system requirements: All products run on IBM PC, XT, AT, PS/2, portable and true compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K RAM minimum. Basic Telecom and Editor Toolboxes require 640K.

Borland International 4585 Scotts Valley Drive, Scotts Valley, CA 95066 Telephone: (408) 438-8400 Telex: 172373

Why more than 600,000 programmers worldwide are using Turbo Pascal today

The irresistible force behind Turbo Pascal's worldwide success is Borland's advanced technology. We created a compiler so fast, that Turbo Pascal* is now the worldwide standard. And there are more tools for Turbo Pascal than for any other development environment in the world.

You'll get everything you need from Turbo Pascal and its 5 Toolboxes

Turbo Pascal and Family are all you'll ever need to perfect programming in Pascal.

If you've never programmed in Pascal, you'll probably want to start with Turbo Pascal Tutor* 2.0, and as your expertise quickly grows, add Toolboxes like our

- Database Toolbox[®]
- Editor Toolbox®
- Graphix Toolbox®
- GameWorks® and our newest,
- Numerical Methods Toolbox

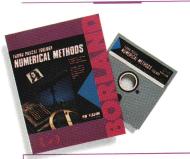


And because Turbo Pascal is the established worldwide standard, 3rd party, independent non-Borland developers also offer an incredible array of programs for Turbo Pascal. *Only \$99.95!*

General Borland International's Turbo Pascal took the programming world by storm. A great compiler combined with a good editor at an astounding price, the package quickly came to be called, simply, Turbo—and has sold more than 500,000 copies.

Stephen Randy Davis, PC Magazine

Language deal of the century. PC Magazine



For Scientists and Engineers: Turbo Pascal Numerical Methods Toolbox

The Numerical Methods Toolbox is a complete collection of Turbo Pascal routines and programs. Add it to your development system and you have the most comprehensive and powerful numerical analysis capabilities—at your fingertips!

The Numerical Methods Toolbox is a state-of-the-art mathematical toolbox with these ten powerful features:

- ✓ Zeros of a function
- ✓ Interpolation
- ☑ Differentiation
- ✓ Integration
- Matrix Inversion
- ✓ Matrix Eigenvalues✓ Differential Equations
- Differential Equati
- ✓ Least Squares
- Fourier Transforms
- ✓ Graphics

Each module comes with procedures that can be easily adapted to your own program. The Toolbox also comes complete with source code. So you have total control of your application.

Only \$99.95!

Turbo Prolog: The Natural Language of Artificial Intelligence

Thether you're a first-time programmer or an experienced one, Turbo Prolog's natural implementation of Artificial Intelligence soon shows you how to build expert systems, natural language interfaces, customized knowledge bases and smart information



Turbo Prolog and Turbo C work hand-in-hand

Turbo Prologº interfaces perfectly with Turbo C* because they're both designed to work with each other.

The Turbo Prolog/Turbo C combination means that you can now build powerful commercial applications using two of the most powerful languages available.

Turbo Prolog's development system includes:

- ☑ A complete Prolog compiler that is a variation of the Clocksin and Mellish Edinburgh standard Prolog.
- A full-screen interactive editor. Support for both graphic and text windows.
- ☑ All the tools that let you build your own expert systems and AI applications with unprecedented ease.

All Borland products are trademarks or registered trademarks of Borland International, Inc., or Borland/Analytica, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Copyright 1987 Borland International

BI-11:

66 An affordable, fast, and easy-to-use language that will delight the newcomer ... You experienced Prolog hackers will likewise be delighted, if not astonished, by the features and performance of the Turbo Prolog development environment.

Turbo Prolog offers generally the fastest and most approachable implementation of that language.

Darryl Rubin, AI Expert 55



How Turbo Prolog's new Toolbox adds 80 powerful tools and 8000 lines of source code

In keeping with Borland tradition, we've quickly added the new Turbo Prolog Toolbox™ to Turbo Prolog.

With 80 tools and 8000 lines of source code that can easily be incorporated into your own programs—and 40 sample programs that show you how to put these AI tools to work—the Turbo Prolog Toolbox is a highly intelligent, high-performance addition. Only \$99.95!

Turbo Prolog Toolbox features include:

- ☑ Business graphics generation: boxes, circles, ellipses, bar charts, pie charts, scaled graphics
- Complete communications package: supports XMODEM protocol
- File transfers from Reflex, dBASE III, 1-2-3, Symphony
- ☑ A unique parser generator: construct your own compiler or query language
- Sophisticated user-interface design tools
- Contains 40 example programs
- your screen layout and I/O
- Calculated fields definition
- Over 8,000 lines of source code you can incorporate into your own programs

The most pow comp

ur new Turbo C generates fast, tight, productionquality code at compilation speeds of more than 13,000 lines a minute!

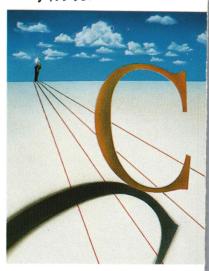
It's the full-featured optimizing compiler everyone has been waiting for.

Switching to Turbo C, or starting with Turbo C, you win both ways

If you're already programming in C, switching to Turbo C will make you feel like you're riding a rocket instead of pedaling a bike.

If you're never programmed in C, starting with Turbo C gives you an instant edge. It's easy to learn, easy to use, and the most efficient C compiler at any price.

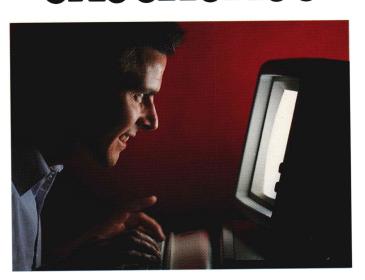
Only \$99.95!



1 Turbo C does look like What We've All Been Waiting For: a full-featured compiler that produces excellent code in an unbelievable hurry . . . moves into a class all its own among fullfeatured C compilers . . . Turbo C is indeed for the serious developer . . . One heck of a buy—at any price.

Michael Abrash, Programmer's Journal 55

Turbo C, Turbo Basic, Turbo Pascal and Turbo Prolog: technical excellence



Borland International's Turbo Pascal, Turbo Basic and Turbo Prolog automatically identify themselves, by virtue of their 'Turbo' forenames, as superior language products with a common programming environment. The appellation also means to many PC users a 'must have' language. To us Turbo C looks like a coup for Borland.

Garry Ray, PC Week*

Garry Ray, PC Week

**Turbo Basic and Turbo Ba

*More Magiç from Blaise. Turbo C TO

Magic is easy with Turbo C TOOLS in your bag of tricks. New Turbo C TOOLS™ from Blaise Computing is a liberary of compiled C functions that allows you full control over the computer, the video environment, and the file system, and gives you the jump on building programs with Borland's new C compiler. Now you can concentrate on the creative parts of your programs.

The library comes with well-documented source code so that you can study, emulate, or adapt it to your specific needs. Blaise Computing's atention to detail, like the use of function prototyping, cleanly organized header files, and a comprehensive, fully-indexed

C TOOLS

SER REFERENCE MANUAL

manual, makes Turbo C TOOLS the choice for experienced software developers as well as newcomers to C.

Turbo C TOOLS provides the sophisticated, bullet-proof capabilities needed in today's programming environment, including removable windows, "sidekickable" applications, and general interrupt service routines written in C.

The functions contained in Turbo C TOOLS are carefully crafted to supplement Turbo C, exploiting its strengths without duplicating its library functions. As a result you'll get functions written predominantly in C, that isolate hardware independence, and are small and easy to use.

Turbo C TOOLS embodies the full spectrum of general purpose utility functions that are critical to today's applications. Some of the features in Turbo C TOOLS are:

- WINDOWS that are stackable and removable, that have optional borders and a cursor memory, and that can accept user input.
- **♦ INTERRUPT SERVICE ROUTINE sup**port for truly flexible, robust and polite applications. We show you how to capture DOS critical errors and keystrokes.
- ◆ INTERVENTION CODE lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple function calls, you can schedule a Turbo C function to execute either when a "hot key" is pressed or at a specified time.
- RESIDENT SOFTWARE SUPPORT lets you create, detect, and remove resident utilities that you write with Turbo C TOOLS.
- T DIRECT VIDEO ACCESS for efficiency, and support for all monitors including EGA 43-line mode.
- **♦ DIRECTORY AND FILE HANDLING** support let you take advantage of the DOS file structure, including volume labels and directory structure.

In addition to Turbo C TOOLS, Blaise Computing Inc. has a full line of support products for Microsoft, Lattice

and Datalight C, Microsoft and Turbo Pascal. Call today for details, and make magic!

BLAISE H

OLS PLUS \$99.95

Screen and window management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more. For Turbo Pascal.

Turbo POWER SCREEN

#include <bwindow.h>

WIN_NODE **

WIN_NODE

uncover (pr

*pnode!

*pcorr

COMING SOON! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo ASYNCH PLUS

Interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem and XMODEM control. For Turbo Pascal.

PASCAL TOOLS/TOOLS 2 \$175.00

Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

Windows; ISRs; screen handling; multiple monitors; EGA 43-line text mode; direct screen access; DOS file handling and more. For MS and Lattice C version 3.00 and later.

Windows; ISRs; EGA 43-line text mode; direct screen access; DOS file handling and more. For the Datalight C compiler.

Full featured interrupt driven support for the COM ports. 1/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM. For C or MS-Pascal.

\$275.00

General screen control; paint screens; block mode data entry or field-by-field control with instant screen access. For C or MS-Pascal.

Text formatter for all programmers; flexible printer control: user-defined variables; index generation; general macro facility. Crafted in Turbo Pascal.

\$95.00

NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

Turbo C is a trademark of Borland International.

t Pascal		_ copies o	1
YES! I want to	make magic!	i a on your proc	as on for
YES! I want	me more informa	mestic orders add	rd air.
☐ Please send	add Sales Tax. Do	ral Express standar)
CA residents	g, \$10.00 for red	ation on your production on your production on your production or your production on your production of your production on your production of your	Zip:-
Name:		_State:	Exp. I
Address.—			
City: VISA or N	1C#:		
VISA		Turbo C	is a trade

BLAISE COMPUTING INC.

Turbo C

TOOLS

supports the Borland

Turbo C compiler, requires DOS 2.00 or

\$129.00

later and is just

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441 CIRCLE 159 ON READER SERVICE CARD

Turbo Basic introduces its powerful new Telecom, Editor and Database Toolboxes

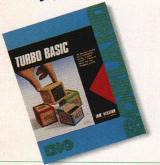
urbo Basic® is the breakthrough you've been waiting for. The same power we brought to Pascal with Turbo Pascal has now been applied to BASIC with Turbo Basic.

Compatible with BASICA, Turbo Basic is the high-performance, high-speed BASIC you'd expect from Borland.

Basically, Turbo Basic is all you need

It's a complete development environment which includes an incredibly fast compiler, an interactive editor and a trace debugging system. It outperforms all its rivals, and because it's compatible with BASICA, you probably already know how to use it.

Includes a free MicroCalc" spreadsheet complete with source code. Only \$99.95!



A technical look at Turbo Basic

- Full recursion supported
- ✓ Standard IEEE floating-point format
- Floating-point support, with full 8087 (math co-processor) integration. Software emulation if no 8087 present
- Program size limited only by available memory (no 64K limitation)
- ▼ VGA, CGA, and EGA support
- Access to local, static, and global variables
- ▼ Full integration of the compiler, editor, and executable program, with separate windows for editing, messages, tracing, and execution
- ☑ Compile, run-time, and I/O errors place you in the source code where error occurred
- ✓ New long integer (32-bit) data
- Full 80-bit precision
- ☑ Pull-down menus

66 Borland has created the most powerful version of BASIC ever.

Ethan Winer, PC Magazine



Telecom Toolbox is a complete communications package which takes advantage of the built-in communications capabilities of BASIC—use as is or modify.

- · Pull-down menus and windows
- XMODEM support
- VT 100 terminal emulation
- Captures text to disk or printer
- PhoneBook file
- 300, 1200, 2400 baud support
- Supports script files
- Fast screen I/O
- · Supports most of XTalk's command set
- Manual dial and redial options

Use Telecom Toolbox to embed communications capabilities into your own programs and/or build your own communications package. Source code included for all Toolbox code and sample programs. Only \$99.95!

For the dealer nearest you or to order by phone call

(800) 255-8008

in CA (800) 742-1133 in Canada (800) 237-1136



SUMMER BREAK SPECIAL! Buy Turbo Basic and get a FREE product. See your dealer for details!

CIRCLE 161 ON READER SERVICE CARD

Database Toolbox means that you don't have to reinvent the wheel each time you write new Turbo Basic database programs.

NEW!

- "Trainer" shows you how B+ trees work. (Simply key in sample records and you'll see your index being built.)
- Turbo Access instantly locates, inserts or deletes records in a database—using B+ trees.
- Turbo Sort sorts data on single items or on multiple keys and features virtual memory management for sorting large data files.

Source code included.

Only \$99.95!



Editor Toolbox is all you need to build your own text editor or word processor. Includes source code for two sample editors.

First Editor is a complete editor ready to include in your programs, complete with windows, block commands and memory-mapped screen routines.

MicroStar" is a full-blown text editor with a complete pull-down menu user interface, and gives you

- Wordwrap
- Undo last change
- Auto-Indent
- · Find and Find/Replace with options
- Set left/right margins
- · Block mark, move and copy
- · Tab, insert, overstrike modes, line center etc.

Includes source code.

Only \$99.95!

ARTICLES

Circle algorithms	How Many Ways Can You Draw a Circle?	18
	by James F. Blinn The fine art of circular reasoning, with numerous example	10
Comparing files	A Survey of File Comparison Algorithms	28
	by Tom Steppe	
	Tom examines various alternative approaches in making a good file comparison utility and follows with C source for own offering.	
Linking lists	The XOR Chain Revisited	36
	by Bennette R. Harris	
	Here's how to use the exclusive-OR operation to compress pointers into a single link field of a doubly linked list.	two
pos drivers	Writing MS-DOS Device Drivers in C	44

COLUMNS

Andy uses an example print driver written in C to show how

Keeping time

C CHEST

106

by Allen Holub

100

Allen examines the use of the PC system clock and DOS interrupts in a metronome program he's written for a MIDI application.

Quad trees

STRUCTURED PROGRAMMING by Namir Clement Shammas

to write installable device drivers.

122

Namir's column begins with a look at V.I.P. (Visual Interactive Programming), an icon-based programming environment for the Mac. He also offers algorithms for optimizing searches and sorts of clustered binary trees.

Smalltalk >

ARTIFICIAL INTELLIGENCE

128

by Ernest R. Tello

Ernie takes a look at Digitalk's Smalltalk/V for the PC.

FORUM		PROGRAMMER'S SERVICES	
EDITORIAL by Tyler Sperry RUNNING LIGHT by Tyler Sperry ARCHIVES LETTERS by you SWAINE'S FLAMES by Michael Swaine	6 8 8 12 152	ADVERTISER INDEX: 13 Where to go for more information on products OF INTEREST: 14 Products for programmers	



About the Cover

It's a problem we've all faced, and it dates back to Og, the caveman. You struggle with primitive tools and hostile environments, only to discover that someone else has already invented the wheel. Special thanks to Bob Wynne, our production manager, for battling saber-toothed tigers every month to put this magazine in your hands-and for taking the time to model for the cover photo. All through the photo session, we heard him muttering to himself, "What goes round, comes round

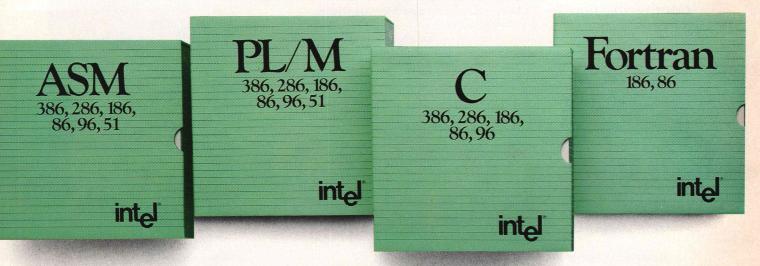
This Issue

In case you hadn't guessed, it's algorithm time again. We've got a good selection of items for your toolbox: everything from going in circles to climbing trees. Now all you need is a generic data structure (one size fits all).

Next Issue

October's DDJ is our annual Forth issue. Among the articles is a demonstration of how threaded subroutines let a 68000 Forth approach the speed of compiled languages. We also have a special feature on hacking the AppleTalk network to add remote nodes via an ordinary RS-232 serial link.

WRITE FASTER IN ANY LANGUAGE.



If you develop software for any product based on an Intel microcontroller or microprocessor, including the 80386, the unique debug hooks in the Intel languages will help get the job done faster.

In fact, when used with Intel debuggers and emulators, Intel development languages can provide more debug data than any other high-level language.

Debug hooks let you

symbolically debug in the same high-level language you wrote in without having to deal with machine or hex code. Which means 80.386 reads as 80.386, not 50 62 D0 C5.

Because the location of both code and data are easily specified with our locator, it is easier for you to develop ROM-based firmware.

Since Intel languages

produce identical object code regardless of the host, you can write code at a PC running DOS, a VAX*/ VMS terminal, or an Intel

Development System.

Software Debuggers
PMON 386, Pscope 86

intel

Different members of the same

design team can therefore choose the most effective combination of languages and systems to get the job done faster.

Intel post-sales support can also help you get the job done faster. We invented the microprocessor. We know microprocessors and languages for Intel architectures better than anyone else.

When you buy an Intel language, you have access to our customer hotline. So if you ever have a question you can talk directly to a trained

Utilities
Relocator, Linker,
Librarian

AEDIT
Programmers'
Editor

intel

intel

applications specialist who understands our products. And can give you the right answers. Faster.

To order today, or get more information—including a free catalog of our development tools—call toll-free 1-800-87-INTEL.

The sooner you call, the faster you'll get the job done.



© 1986 Intel Corporation *VAX is a registered trademark of Digital Equipment Corporation.

EDITORIAL

Stone Age Computing

This month's DDJ has, without argument, one of the most thought-provoking covers we've produced. The mix of Og, the caveman, and a pseudocoded cave wall is incongruous enough to provoke a hoard of questions in the reader's mind. What does that code on the wall really say? Why is Og staring so intently at the bottom line of the algorithm? And perhaps most important of all: has the staff at DDJ finally gone round the bend?

Once you get past the fun captions and lines about "recoding the wheel," however, there's an aspect of Og's situation that is jarringly similar to the present day. Simply put: today's personal computers often seem (despite their sexy new processors and sophisticated designs) to be the product of Neanderthal thinking. Again and again we're presented with products that are inexplicably primitive or crippled.

Apple's Macintosh, for example, seems to be a curious mix of high-tech and high-Bronze Age. Much of the design work in the Macintosh is wasted in a machine aimed at people only slightly more sophisticated than our friend Og. We have icons for all those computer users who are too stupid or lazy to read English and a single-button mouse for those unable to handle a keyboard. To its credit, Apple has recently added an innovation for the Mac's most advanced users—the keyboards now have cursor keys.

The problems of primitive thinking and design are hardly an Apple monopoly, of course. Microsoft's MSDOS, based on ideas from CP/M and Unix, is so primitive that it defies tasteful jokes. On the other hand, OS/2—with over a million lines of code yet to be debugged by adventurous users—has tremendous potential for being as good a running gag as Microsoft's Windows. Assum-

ing, of course, it actually runs.

And then there's the Intel architecture.... Has anyone out there found a solution to switching back and forth between protected and real mode in the 80286? Being able to use both modes was going to be a major feature of the chip, but here we are-years after the introduction-and the accepted method of switching modes is to reset the processor. The method used by Microsoft in OS/2 (a reset request via the keyboard controller) seems to be the best kludge so far. That is, it's painfully slow, but it works. You'd think there'd be something less, well, "Stone Age" than putting the chip to sleep for a couple of milliseconds.

In flaming, it's generally a good idea to keep your historical perspective. This editorial was partially inspired by a letter from one of our readers, John Dvorak. Always eager to stir things up, John questioned our ability to be nasty and rude. This editorial could be seen as a reply. Flaming is certainly a fun way to fill space in a magazine, but at DDJ we try to remember that the purpose is to shed light, not just to blow smoke. Or, as Mrs. Og said when she saw the wheel for the first time, "That's very nice, dear. But what's it for?"

Tyler Sperry editor

Dr. Dobb's Journal of Software Tools

Editorial

Editor-in-Chief Michael Swaine

Editor Tyler Sperry

Managing Editor Vince Leone
Associate Editor Bon Copeland

Associate Editor Ron Copeland
Assistant Editor Sara Noah Ruddy

Technical Editors Allen Holub Richard Relph

Contributing Editors Namir Shammas

Ernest R. Tello

Copy Editor Rhoda Simmons

Production

Production Manager Bob Wynne
Art Director Michael Hollister

Assoc. Art Director Joe Sikoryak
Technical Illustrator Frank Pollifrone

Typesetter Mary Lopez

Cover Photographer Michael Carr

Circulation

Circulation Director Maureen Kaminski
Book Marketing Mgr. Jane Sharninghouse
Subscription Supervisor
Newsstand Sales Larry Hupman

Administration

Finance Director Kate Wheat
Business Manager Betty Trickett
Accounts Payable Supv. Mayda Lopez-Quintana

Accts. Receivable Supv. Laura DiLazzaro
Advertising Director

Ferris Ferdon (415) 366-3600 Account Managers see page 137 Promotions/Srvcs. Mgr. Anna Kittleson

Advertising Coordinator Donna Rogers
Associate Publisher
Michael Swaine

Assistant Sara Noah Ruddy

Dr. Dobb's Journal of Software Tools (USPS 307690) is published monthly by M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points. DDJ is published under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.

Article Submissions: Send manuscripts and disk (with article and listings) to the Editor.

DDJ on CompuServe: Type GO DDJ

Address Correction Request: Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128.

Customer Service: For subscription problems call: outside CA (800) 321-3333; in CA (619) 485-9623 or 566-6947. For book/software order problems call (415) 366-3600.

Subscriptions: \$29.97 per 1 year; \$56.97 for 2 years. Canada and Mexico add \$27 per year airmail or \$10 per year surface. All other countries add \$27 per year airmail. Foreign subscriptions must be prepaid in U.S. funds drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

Foreign Newsstand Distributor: Worldwide Media Service Inc., 386 Park Ave. South, New York, NY 10016; (212) 686-1520 TELEX 620430 (WUI).

Entire contents copyright © 1987 by M&T Publishing, Inc., unless otherwise noted on specific articles. All rights reserved.



M&T Publishing Inc.

Chairman of the Board Otmar Weber
Director C.F. von Quadt
President and Publisher Laird Foshay

Our thanks to NASA for supplying this computer enhanced ultraviolet photo taken by Skylab IV of a solar prominence reaching out 350,000 miles above the sun's surface

Genius Begins With A Great Idea

But The Idea Is Just The Beginning

What follows is the time consuming task of giving form and function to the idea.

That's why we concentrate on building into our software development systems functions and features that help you develop your software ideas in less time and with less effort.

We've started 1987 by releasing new versions of our MS-DOS, Macintosh, Amiga, ROM, and Apple // C development systems. Each system is packed with new features, impressive performance, and a little bit more genius.

Aztec C86 4.1 New PC/MS-DOS • CP/M-86 • ROM

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

Aztec C86-p Professional System \$199 · optimized C with near, far, huge, small, and large memory + Inline assembler + Inline 8087/80287 + ANSI support + Fast Float (32 bit) + optimization options • Manx Aztec 8086/80x86 macro assembler · Aztec overlay linker (large/small model) · source level debugger • object librarian • 3.x file sharing & locking . comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System \$299 includes all of Aztec C86-p
 Unix utilities make, diff, grep • vi editor • 6 + memory models • Profiler.

Aztec C86-c Commercial System. \$499 • includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

Aztec C86 Third Party Software

A large array of support software is available for Aztec C86. Call or write for information. The following is a list of the most requested products: • Essential Graphics • C Utility Library • Curses • Greenleaf Communication, General, and Data Window • Halo • Panel + • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data • C terp • db_Vista • db-Query • Phact • Plink-86 Plus • c-tree • r-tree • Pmate

CP/M • TRS-80 • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Aztec C II-c	(CP/M-80	&	R	0	M).			 .\$349
Aztec CII-d									
Aztec C80 (

Aztec C68k/Am 3.4 **New Amiga Release**

Amiga user groups across the USA voted Aztec C68k/Am release 3.3 the best Software Development System for the Amiga. Release 3.4 is more impres-

Aztec C68k/Am-p Professional\$199 A price/feature/performance miracle. System includes: optimized C . 68000/680x0 assembler . 68881 support • overlay linker • UNIX and Amiga libraries • examples.

Aztec C68k/Am-d Developer \$299 The best of Manx, Amiga, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep and vi.

Aztec C68k/Am-c Commercial \$499 Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C68k/Mac 3.4 **New Macintosh Release**

For code quality, reliability, and solid professional features, Aztec C for the Macintosh is unbeatable. This new release includes features and functions not found in any other Macintosh C development system.

Aztec C68k/Mac-p Professional \$199 optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Macintosh libraries • examples.

Aztec C68k/Mac-d Developer......\$299 The best of Manx, Macintosh, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep • vi editor.

Aztec C68k/Mac-c Commercial \$499 Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C65 New ProDOS Release

Aztec C65 is the only commercial quality C compiler for the Apple II. Aztec C65 includes C compiler, 6502/65C02 assembler, linker, library utility, UNIX libraries, special purpose libraries, shell development environment, and more. An impressive system.

Aztec C65-c Commercial \$299 • runs under ProDOS • code for ProDOS or DOS 3.3 Aztec C65-d Developer..... • runs under DOS 3.3 • code for DOS 3.3

Aztec ROM Systems

6502/65C02 • 8080/Z80 • 8086/80x86 • 680x0

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under \$1,000 outperform systems priced at over \$10,000.

Initial Host Plus Target						.\$750
Additional Targets						.\$500
ROM Support Package						.\$500

Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

C' Prime PC/MS-DOS • Macintosh Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as \$500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more. Special discounts are available for use as course material.

C' Prime\$75

Aztec Cross Development Systems

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

How To Become An Aztec C User

To become a user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, MasterCard, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

Aztec C is available directly from Manx and from technically oriented computer and software stores. Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee.

Most systems are upgradable by paying the difference in price plus \$10. Site licenses, OEM, educational, and multiple copy discounts are available.

1-800-221-0440 In NJ or international call (201) 542-2121 • TELEX: 4995812

To order or for more information call today.

MS is a registered TM of Microsoft, Inc. CPIM TM DRI, HALO TM Media Cybernetics, PANEL TM Roundhill Computer Systems, Ltd., PHACT TM PHACT Assoc., PRE-C, Plink-86, Plink-86 + , P-Foce TM Phoenix, db Vista TM Raima Corp., C-terp, P-C-lint, TM Gimpel Soft-ware. C-tree TM Faircom, Inc., Windows for C, Windows for DAN TM Creative Solutions, Apple II, Macintosh TM Apple, Inc., TRS-80 TM Radio Shack, Amig TM Commodore Int.], Unit TM AST, Vax TM DEC, Aztec TM Manx Software Systems.

CIRCLE 108 ON READER SERVICE CARD Manx Software Systems 1 Industrial Way Fatontown, NJ 07724

RUNNING LIGHT

Tam often struck by the similarity between editing a magazine and the plight of Billy Pilgrim in Slaughterhouse Five. Vonnegut's hero, you may recall, had become "unstuck in time" and began to experience the moments of his life—both past and

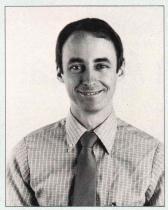
future—out of the normal sequence. A day spent in old age might be followed by a week of his youth, the shift coming without any apparent plan or warning.

Editing a magazine is a lot like that.

From the writer's perspective, things are pretty straightforward. You write and debug your program, write an article about it, and send it in to *DDJ*. After an awfully long time, you get a letter telling you if we liked the manuscript enough to print it.

The view from the editor's desk is a lot different. As I write this column, it's the middle of July, and this piece is literally the last thing to go into the magazine before we ship it out the door in preparation for printing. As soon as I finish this column, there are neglected Forth articles to edit for the October issue. All this happens when I should be soliciting authors and articles for the December operating systems issue or at least checking on the progress of the authors and editors working on stuff for the November graphics issue. All this high-speed flipping through the calendar is enough to drive a normal person quite mad. Having spent so much time with computers, of course, I am already certifiable.

I mention all this chaos, not in an appeal for sympathy—although it would be nice if they'd quit sticking pins in the voodoo doll—but so that you'll understand why it sometimes



takes months to evaluate a manuscript. It's the old business of trying to drain the swamp when the alligators keep demanding your attention. Sometimes a week will pass and I'll suddenly realize I haven't looked at the latest batch of manuscripts, or checked

into the DDJ forum on CompuServe. Worse, even after we've considered a piece, there's a good chance it will have to be sent off to a referee or Technical Editor.

Things are getting better, and we're speeding up the evaluation cycle. Ron Copeland has joined our staff as Associate Editor, and has already been an immense help both in tracking down manuscripts that were lost on my desk and in routing them to the proper referees. Richard Relph, who coordinated our infamous giant C Compiler roundup two years running and has written a number of articles for us, has just joined the staff as a Technical Editor. By the time you read this, our response time on manuscripts should have dropped to less than a month.

So, to those of you who got caught in the old delay loops, my apologies. For those of you who haven't sent in an article or query, there's no longer any excuse. If you've done something particularly neat or solved a particularly vexing problem, let our readers know about it by submitting an article. If you want to discuss an article idea with me, give me a call at (415) 366-3600. You can also send me e-mail via CompuServe (76703,4266), or BIX (with the clever nom-de-baud of "tyler").

Tyler Sperry editor

ARCHIVES

Intel Needs Help

"From the January-February issue of Solutions, a public-relations magazine put out by Intel Corp., we learn that Intel has prepared a Support Library for their 8087 numeric co-processor. It 'consists' of the function library, a decimal conversion module, ... and a full 8087 software emulator for debugging without the 8087 component. The library ... supports the proposed IEEE Floating Point Standard.

"Well, that's a lot of software (even if the 8087 does do most of the work), so we suppose that Intel's asking price of \$1,250 (gulp!) is justified. But is the 8087 so hard to work with that you need twelve hundred dollars worth of interface code? Surely not! Obviously, Intel doesn't know how to sell chips. Let's show them how it's done: let's put some 8087 knowledge into the public domain so that experimenters can use the device. Many readers working with it? What kind of programming does it take?"—D. E. Cortesi, Dr. Dobb's Clinic, DDJ, June 1982.

Ten Years Ago in DDJ

"OLDIE BUT GOODIE AVAILABLE

"Our glorious(?) Editor spent much of a decade (no pun intended) in mutually supportive relationships with the 8-Family of Maynard, and has long had a warm spot in his heart for these quaint machines. (For the younger generation and the uninitiated, this eccentric discussion concerns one of the first—and, for many years, the most popular—minicomputers: the PDP-8 manufactured by Digital Equipment Corporation (DEC) in Maryland, MA.)

"Back in the Summer of '76, the blossoming of DDJ, the Computer Faire, the Silicon Gulch Gazette, the ACM, teaching at Stanford, NCC in Dallas, PC'77.... and the list goes on and on. And our Editor has had little time to do more than turn his old friend on from time to time, and watch it glow and blink. Lest this old 8/1 feel spurned and lonely—for it is an honorable machine with a strong Puritan work ethic, not accustomed to having its diodes idle—Jim is seeking a good home for it. Price for the whole system FOB Menlo Park, California is \$4,000."—Jim Warren, DDJ September 1977.



The Fastest C

We challenged Microsoft-C to a C compiler duel, measuring compile, link, and execution times. They wouldn't take on Optimum-C — they own it and they know how fast it is!

DATALIGHT's new Optimum-C compiler can make your programs run up to 30% faster than other compilers. That's because it's a true optimizing compiler. While many manufacturers may claim to have optimizing compilers, they can't stand up against Optimum-C. Their optimizers look at only one or maybe a few statements at a time; Optimum-C takes in the BIG picture, looking at an entire function at once.

Optimum-C tunes each function to the machine by maximizing register usage while minimizing memory usage. Optimizations performed include: global common subexpression elimination, register allocation by coloring, copy/constant propagation, loop induction variables, and dead code elimination.

Develop Programs Faster

To save compile time during code development, you can turn off the optimization. Then, when your program is developed and debugged, you can turn on the optimization and recompile for the fastest execution speed.

Also included to help you with code development is the EZ interactive editor/environment. With EZ you can compile, link, execute and quickly correct any syntax errors by letting EZ show you the erroneous source line and error message.

Library Source Code and Other Extras

Top speed is only one of the advantages you get with Optimum-C. You also get complete library/startup source code (at no additional charge) support for 4 memory models (including large model), inline 8087 code, and standard object files. Plus a complete UNIX style MAKE program and a fast screen I/O package.

Speed ROM Development

It used to be that to develop ROM code you had to purchase a compiler from one manufacturer, a debugger from another, and other tools from still another. DATALIGHT has eliminated all the hassle by providing all the support tools necessary for ROM development.

BENCHMARKS

	Sieve	Pointer	Optimize
Optimum-C	49.3	45.7	00.9
MS C	58.6	90.2	38.6
Turbo-C	62.1	49.0	40.2

ROM-it, the Unique ROM Development Kit

Only DATALIGHT offers you the kind of support you'll find in Rom-it, the new ROM support package. This unique package speeds up the delivery of your ROM application by providing many program elements that you used to have to write for yourself.

Rom-it includes a startup routine that can take an 8086 from restart to C program execution by setting up segment registers, stack, heap, copying initialized data from ROM to RAM, and zeroing uninitialized data. The BLAZE locator/Intel hex file generator can locate code and data anywhere in memory, while performing checks for ROM overruns, illegal data access, and complete program location. The ROMable library contains the ROMable portions of the Optimum-C library, with support for interrupt handling in C, reading and writing I/O ports, and full math support.

Debug ROMed Code from the PC

You can download, execute, and debug programs on the target system from your PC, with Remote-DSD. This debugger is a symbolic windowing debugger that shows you the C source code (on the PC) that is executing on the target system. You can view and change global variables on the target system, modify data, set breakpoints and watch registers and flags.

Try Optimum-C risk free

Call us TOLL FREE today for your copy of Optimum-C. You will also receive free* "C: A Programming Workshop" to get you started with C. To find out how we can help you get your ROM projects completed on time call us.

Optimum-C w/ C Tutorial \$139 Rom-it \$159

Add \$7 for shipping in US/\$20 outside US COD (add \$2.50)

Not Copy Protected

ORDER TOLL-FREE TODAY!

1-800-221-6630

ATTENTION OEMs!

Contact us regarding arrangements.

*Limited offer available exclusively to readers who purchase directly from Datalight.

Microsoft and MS-DOS are registered trademarks of the Microsoft Corporation. Turbo C is a registered trademark of Borland International.

Magazine Reviewers Shocked by DATALIGHT's Performance...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavyweights." Datalight C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

DR. DOBBS, August 1986

"This is a sharp compiler!... what is impressive is that Datalight not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

COMPUTER LANGUAGE, February 1986

Optimum-C Version 3.08

NEW!

EZ Interactive Development Environment

NEW!

Inline 8087/80287 Math Support

- Full UNIX System 5 C language plus ANSI extensions
- Fast/tight code via powerful optimizations including common subexpression elimination
- DLC one-step compile/link program
- Multiple memory model support
 UNIX compatible library with PC
- UNIX compatible library with PC functions
 Compatible with DOS linker and
- assembler

 Third-party library support
- Automatic generation of .COM files
- Supports DOS pathnames, wild cards, and Input/Output redirection
- Compatible with Lattice C version 3.x
- Interrupt handling in C
- Debugger support
- ROMable code support/start-up source
- Full UNIX MAKE Program
- Tools in Source Code: cat, diff, ...

MS-DOS® Support Features

- Mouse support
- Sound support
- Fast screen I/OInterrupt handler

Datalight

17505-68th Avenue NE, Suite 304 Bothell, Washington 98011 USA (206) 367-1803

On April 2,1987 IBM and Quarterdeck announced the next generation in personal computing.

Introducing DESQview 2.0. Improving the past and ready for the future right now.

In one sweeping announcement from Miami Beach and New York City, IBM established new standards of performance for personal computers, with its new Personal System/2.™ Quarterdeck was there with IBM and simultaneously helped establish new standards for multi-tasking and multi-windowing.

We were there for them then. We're here for you now.

If you use two or more software programs, if you use a PC-compatible machine, if you own a new 80386 computer, if you've just bought one of the new Personal System/2 computers, or if you've tried Microsoft Windows and were disappointed but still need the power of graphics programs, DESQview 2.0 is the answer.

Consider this. InfoWorld voted DESQview's earlier version 1986 Product of the Year. SoftSector gave it the Editor's Choice Award. In PC Tech Journal's "System Builder Contest" at Comdex Fall it was voted best operating environment. And 450,000 dedicated users on four continents have voted yes with their dollars.

The new DESQview 2.0 is an order of magnitude better.

This unique software program enhances the power of your personal computer and makes it more convenient to use. It still gives your PC the power of many PCs. It still does windows. It still multi-tasks. It still breaks the DOS 640K barrier. It still transfers data. It still dials your phone. It still gives you menus for DOS. It still remembers your keystrokes (macros). It still runs your existing programs and your new programs soon to come. In fact now you can even run Windows-, GEM-, and Topview-specific programs too. And with 386 machines and our Expanded Memory Manager it still becomes a 386 control program, but now you can run text and CGA graphics programs in background.

The new DESQview 2.0.

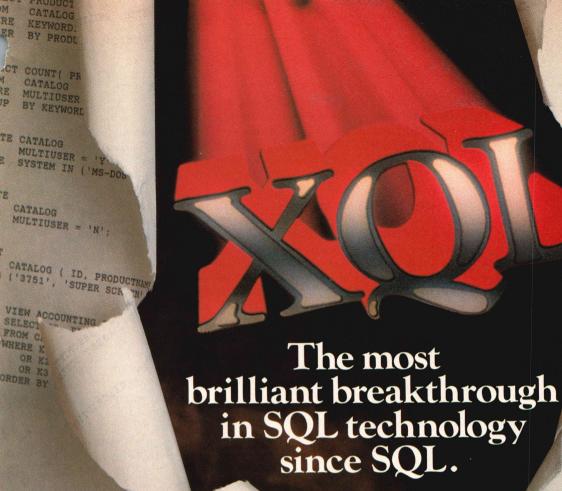
For us it's the next logical step. For you it's windows of opportunity. SYSTEM REQUIREMENTS

 IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286 or 80386 processors) with monochrome or color display; IBM Personal System/2 · Memory: 640K recommended; for DESQview itself 0-145K • Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage · Disk: Two diskette drives or one diskette drive and a hard disk · Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA) · Mouse (Optional): Mouse Systems, Microsoft and compatibles . Modem for Auto-Dialer (Optional): Hayes or Compatible Operating System: PC-DOS 2.0-3.3; MS-DOS 2.0-3.2 · Software: Most PC-DOS and MS-DOS application programs; programs specific to TopView 1.1, GEM 1.1 and Microsoft Windows 1.03 · Media: DESQview 2.0 is available on either 51/4" or 31/2" floppy diskettes

		DEGG		1000	Retail P		Total	
	18 70 F 18 18 18 18 18 18 18 18 18 18 18 18 18	DESG	view 2.0	Arrive Ser	\$129.9	5	\$	
	Shipping & Har	ndling	USA Outside	USA	\$ 5.0 \$ 10.0	-	\$	
	S	ales Tax	(CA reside	ents)	6.5%		\$	
Payment: 🗆 V	√isa □ MC □	AMEX	☐ Check			ount losed	\$	
Credit Card: Va	lid Since	/_		Expi	ration _		_/	
Card Number:							1	
Credit Card Name					4.5.3			
Shipping Address								
		04	ate	7in		Tolor	h	



Quarterdeck Office Systems • 150 Pico Boulevard, Santa Monica, CA 90405 • (213) 392-9851



XQL is a dramatic step forward in the history of SQL. It's the one unique SQL solution that helps programmers break through to even higher levels of productivity. Powerful yet easy to use, XQL minimizes your coding time and lets you focus on building better applications.

XQL extends the power of **Btrieve**, SoftCraft's high-performance file manager, by allowing access to multiple records at a time. It frees your application from physical file characteristics by providing true relational capabilities with data independence, data descriptions, data integrity and security.

XQL's three interface levels are a major advance in SQL technology. The first two levels, XQL primitives

for maximum
efficiency or full
SQL statements for
maximum convenience, are callable
subroutines from
BASIC, Pascal and C. The third
level lets you enter SQL statements
interactively without ever having
to write a program.

XQL's extensive DBMS features let you access data by name. Field order is independent of physical location within the Btrieve record. Only records that pass your restrictions are returned—in the sort order you specify. Fields can be computed from other fields or constants. And you can manipulate composite records built from multiple, joined Btrieve files.

the performance and reliability you've come to expect from Btrieve, including LAN support, fault tolerance, comprehensive documentation and expert technical support for trouble-free software development.

Plus, you never pay royalties on your XQL applications.

Put the latest innovation in SQL technology to work for you. Contact SoftCraft.



XQL, \$595; Btrieve, \$245; multiuser Btrieve, \$595. XQL requires Btrieve and PC-DOS or MS-DOS 2.X or 3.X. XQL is a trademark and Btrieve is a registered trademark of SoftCraft, Inc.

LETTERS



Dvorak Responds

Dear DDJ,

I was greatly amused (Swaine's Flames, June 1987) by the rambling cockroach-like insect that supposedly jumped out of Swaine's computer and onto the keyboard late one night while Mike was puking his guts out after having been to one of those dubious and fashionable sushi bars found far too often south of Palo Alto where fresh fish is a rarity and where the wallpaper, when examined closely, turns out to be posted health department notices of violation.

No doubt was left in the reader's mind that the whole incident was a

hallucination of grand proportion or (and we all know that this is the case) the column was a gimmicky idea possibly dreamed up while suffering the ill effects of poisoning and designed to solve one of two problems. They are: 1) how to make a point and blame someone (or something) else for making the point; 2) how to avoid the ravages of mean copy editors. Since the second point only applies to the profession of writing, one must assume that the first point applies.

Now obviously some assertion I made sometime when and who knows where struck a nerve inside the *Dr. Dobb's* editors cage. Thus the essay by the roach about me. Unfortunately, for Mike and the crew, my point concerning outspokenness was only reaffirmed by the technique of using an imagined insect to

express an opinion. It surely wasn't done to avoid being nasty but only done to avoid confrontation. That's the problem (if there is a problem), in my opinion, with *Dr. Dobb's*—a certain inexplicable timidity mistakenly interpreted by some as politeness.

Now I advise you to look at my current benefactor and publisher, *PC Magazine*. The commentary in there is sometimes downright rude. Editors even have the gall to make Editors Choices, a concept that flies in the face of advertising theory. Yet the magazine is as fat as imaginable and growing in circulation like nothing I've ever seen. Why? Because today's reader is bored and prefers to read copy written with a sharp edge. There is too much dreary stuff out there taking up too much space in our lives.

The editors of *Dobb's*, I know for a fact, have this same ability to produce enjoyable vitriol but prefer to be well liked instead. The out-of-place and gratuitous swipe at me by the verbose roach claiming I was an incarnation of Max Headroom is a good example of the kind of nastiness that lurks within the *Dobb's*

staff.

While the entire piece was a gem of creativity insofar as finding some unusual way to couch an essay, I still maintain that using a bug to make a complaint is symbolic not of dignity but of a certain kind of shyness that weakens the impact of the printed word. Then again, the whole story may be true. If so, I advise Swaine to vacuum more often.

John C. Dvorak PC Magazine Ziff-Davis Publishing Co. One Park Ave. New York, NY 10016

Michael Swaine replies:

Thanks, John. It's an honor to get a lesson from the leading practitioner of no-nonsense, outspoken, sharpedged computer journalism. By the way, I enjoyed your recent *PC Magazine* column tracing the origin of the word *nerd* back to Dr. Seuss.

XOR Chains

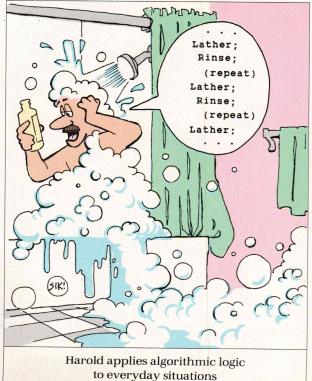
Dear DDJ,

David Cortesi's article on XOR chains (June 1987) is an excellent example of isolating information in modules.

I disagree with one of his conclusions, though: the *Insert* procedure can do the same damage to a list as the *Delete* procedure can. When two different *Scans* each insert an item into the same position in the list, the list structure is destroyed.

For example: let Scans P and Q each have item A as next and item B as prev. Then Insert (C,P) will insert an item C between A and B in the list. Scan P will correctly show item A as next and item C as prev, but Scan Q will still treat items A and B as adjacent items. Calling Insert (D,Q) now will cause Insert to try. incorrectly, to XOR B to the link in A as if they were still adjacent items. This XOR operation and the XOR of A to the link in B will leave A and B with invalid links.

This flaw in *Insert* can be corrected by having *Insert*



To Develop Tomorrow's Applications, You Need The DOS Of The Future.



THE MULTITASKING, MULTIUSER MS-DOS REPLACEMENT

- RUNS ON ANY IBM-PC, XT, AT, 80386, OR TRUE COMPATIBLE
- RUNS MS-DOS PROGRAMS AND USES THE MS-DOS FILE SYSTEM
- FEATURES FILE SHARING AND FILE LOCKING
- PROVIDES TRUE CONCURRENT MULTITASKING AND TASK SWITCHING
- SUPPORTS MULTIPLE TERMINALS WITH NO EXTRA SOFTWARE REQUIRED
- FEATURES A USER-CONFIGURABLE WINDOWING INTERFACE
- ALLOWS ADDRESSING OF EXTENDED MEMORY
- FEATURES A FILE PERMISSION SYSTEM
- SUPPORTS THE MS-DOS COMMAND LANGUAGE AND EXTENDS IT WITH COMMANDS LIKE PROTECT, PRIV, SPAWN, AND KILL

ONLY \$99 — DISKETTES AND MANUAL INCLUDED

And You Need The Tools To Get The Job Done Right . . .

T.M.

- GIVES ACCESS TO OVER 80 NEW SYSTEM SERVICES SUPPORTED BY WENDIN-DOS
- INCLUDES LIBRARIES WRITTEN IN ASSEMBLY FOR ACCESS FROM HIGH-LEVEL LANGUAGES
- ALLOWS USER PROGRAMS TO ACCESS SYSTEM SERVICES FOR PROCESS CONTROL SWAPPING, MEMORY MANAGEMENT, RECORD AND FILE MANAGEMENT, AND CONCURRENT I/O.
- PROVIDES AN I/O SUBSYSTEM THAT SUPPORTS CONCURRENT I/O OPERATIONS ON ANY OF THREE DIFFERENT LEVELS OF ACCESS: PHYSICAL, LOGICAL, AND VIRTUAL.

ONLY \$99 — DISKETTE, SOURCE CODE AND MANUAL INCLUDED

TO ORDER WENDIN-DOS or THE WENDIN-DOS APPLICATION **DEVELOPER'S KIT — CALL (509) 624-8088 or**

SIMPLY SEND A BRIEF, WRITTEN REQUEST FOR INFORMATION ABOUT ANY WENDIN PRODUCT TO:



Wendin, Inc. P.O. Box 3888 Spokane, WA 99220-3888

We will send you an exciting full color catalog and a FREE product line demo diskette, while supplies last, compliments of Syncom Technologies, Inc., and Wendin, Inc. (509) 624-8088 (System hardware recommendation — minimum 512K and for multitasking a machine with at least the computing power of an IBM-AT.)



BOX 3888 SPOKANE, WA 99220-3888

Working beyond the horizon to develop the operating systems of tomorrow © Copyright 1987 Wendin, Inc. (509) 624-8088

DEALER INQUIRIES WELCOME

DEALEM INVOITNES WELCOME
Foreign orders inquire about shipping.
Domestic orders add \$6.00' 1st item, \$1.00
each additional item for shipping, handling, and
insurance. We accept Visa/MC, American
Express, C.O.D., and Bank Drafts drawn on
U.S. Banks.

Washington residents add 7.8% sales tax

MS is a trademark of Microsoft. PC-DOS is a trademark of IBM

Wendin is a registered trademark of Wendin, Inc. Wendin-DOS and Wendin-DOS Application Developers Kit are trademarks of Wendin, Inc.

LETTERS

(continued from page 12)

check if the elements in the *Scan* are still adjacent before inserting the new item.

Andrew Ginter 208-21 Ave. NW Calgary, AB Canada T2M 1J3

Dear DDJ.

In his article Mr. Cortesi mentions clearing a register in assembly language by XORing it with itself. It also might be worth mentioning that you can swap two fields quickly without using a work area via use of the XOR command thrice:

XC FIELD1,FIELD2 XC FIELD2,FIELD1 XC FIELD1,FIELD2

This will put the contents of *FIELD1* into *FIELD2* and vice versa. Note that the fields must be of equal length. I'm sure that this also works just as quickly in other languages because typically Boolean algebra commands are built-in commands in almost any processor.

Jack Gillette 30 Harvest Rust Ct. St. Charles, MO 63303

Dear DDJ.

David E. Cortesi's fine article presented your readers with some valuable information. However, the algorithms he describes are not the simplest possible because he uses an unnecessarily complicated logical tautology. The relation he used in his article was:

(A XOR B XOR C) XOR B XOR C = A

This relation is actually one case of a more general relationship, namely:

I'm not sure if this general relation is of much value, but its simplest form is more suitable for the task of storing both front and back pointers in a single field. The simplest form merely says:

(A XOR B) XOR B = A

Nowhere in the algorithms that

Mr. Cortesi describes does he require the fact that his link field stores the address of the current item in the list. With this in mind, his implementation can be improved with the following changes:

1. For an item in the list, if A is the address of its predecessor and B is the address of its successor, then set the single link word W of the item to:

W = A XOR B

You can now recover B as (W XOR A) or A as (W XOR B).

2. The stepping routines *GoFwd* and *GoBak* should be modified to remove the use of the current item address. For example:

Void GoFwd(s) struct Scan *s; struct Item *i; i = s->prior Xor s->next->link; s- >prior = s->next; s- >next = i;

3. The insertion and deletion routines *Insert* and *Delete* should be modified in a similar fashion. For the sake of accuracy (and to correct some minor errors in the original article), I have given each of these in full in Example 1, right.

The changes described in Example 1 amount to a savings of one XOR operation per call to any of these four routines. Although this may not seem like much, it can make a significant difference in the time required to process a large list. Of course, the effect on code size is minimal.

Mr. Cortesi correctly asserts that traversing a data structure linked in this fashion requires the addresses of two logically adjacent items. However, this does not lock you into a listtype structure. I am submitting a follow-up article entitled "The XOR Chain Revisited" that describes how such links can be used effectively in manipulating tree-type data structures.

Bennette R. Harris 231 S. Janesville St. Whitewater, WI 53190

Mr. Harris' article begins on page 36 in this issue. —eds.

68000 Dynamic Halt

Dear DDJ,

The FOR TOP to BOT loop in Brian Anderson's Viewpoint (June 1987) will only reach BOT in the unlikely event that TOP = BOT! Else we have yet another fine Dynamic Halt (aka endless loop). CLR.B 0(A0,D0) does not alter value y D0. So did your typesetter omit ADDQ.W *1, D0 before the loop branch? Brian would never do such a thing (and yet his line numbers seem (continued on page 138)

```
void Insert(i.s)
struct Item *i;
struct Scan *s;
if (AtHead(s))
 s->base->head=i;
else
  s->prior->link Xor= (i Xor s->next);
if (AtTail(s))
 s->base->tail=i;
else
 s->next->link Xor= (i Xors->prior);
i->link = s->prior Xor s->next;
s->prior=i;
void Delete(s) struct Scan *s;
struct Item *i:
if (AtTail(s)) return:
i = s - > next - > link Xor s - > prior;
if (AtHead(s))
 s->base->head = i;
 s->prior->link Xor=(s->next Xor i);
if(i == Nil)
 s->base->tail=s->prior;
else
 i->link Xor= (s->next Xor s->prior);
DropItem(s->next):
s->next=i;
```

addresses of two logical- **Example 1:** Modified insertion and deletion ly adjacent items. How-routines for "The XOR Chain"

RESIDENT EXPERT Pop-up Reference Guides...



Try One And Get Our MS-DOS/PC-DOS Guide Absolutely FREE!

THE POP-UP REFERENCE REVOLUTION BEGINS

How much development time could you save if you never had to open another PC language or technical reference manual again? What if you could just point at a compiler keyword, assembly instruction, or function name on your screen and with a keystroke have complete, authoritative information about language syntax, operands, parameters, examples, and much more?

INTRODUCING THE RESIDENT EXPERT SYSTEM

A growing library of comprehensive, disk resident reference guides about the PC and your favorite PC languages. All available instantly through our unique memory resident pop-up access system.

VIRTUALLY EVERYTHING YOU NEED TO KNOW

Each of our Compiler Reference Guides contains virtually everything you need to know to program with your preferred implementation of your favorite language. Language syntax, all library functions, complier directives, and error codes are thoroughly documented.

Our PC Programmer's Reference Guide documents every PC (and AT) processor instruction and every BIOS and DOS service interrupt. You'll also find tables of keyboard codes, line drawing, ASCII, and IBM character sets, and much more.

THE SPECIALIST'S LIBRARY

Your compiler is unique. That's why our reference guides are *specialized*...each one designed for a particular vendor's language implementation.

Free With Any Purchase!

Our Companion Pop-up Guide To MS-DOS 3.2/PC-DOS 3.3

Limited Time Offer

OUICK DRAW ACCESS SYSTEM

Point-and-shoot...just place the cursor over any term on your screen. Chances are we've got it fully detailed in one of our data bases.

Fully cross indexed...if the instruction or library function you're using isn't quite right, our related topics cross index can help you find a better one.

Multiple volumes on line...you can have one or a dozen of our pop-up reference guides on line...a complete library available instantly.

THE INFORMATION YOU NEED...WHERE YOU NEED IT

Our pop-up shell varies its size and shape dynamically, only taking as much space on your screen as it needs and it *never* covers your working area. You can see your work and our reference data at the *same* time.

A COMPLETE LIBRARY...STILL ONLY A BEGINNING

At Santa Rita, our commitment is to provide the most accurate, extensive selection of PC language reference materials available. If you don't see one of our guides for your favorite language or compiler listed below don't worry, we're probably working on it!

PC Programmer's Reference Guide ... \$59.00 (with Assembly Language Guide)

Borland Turbo C (1.0)	. 59.00
Borland Turbo Pascal (3.0 and below)	. 59.00
(with Graphics & Numerical Methods Toolbox)	
Borland Turbo Prolog (1.1 and below)	. 59.00
(with Prolog Toolbox)	
Lattice C Compiler (3.2 and below)	. 59.00
Mark Williams LetsC (4.0 and below)	. 59.00
Microsoft C Compiler (5.0 and below)	

SantaRita

For the location of your nearest Santa Rita Software dealer, or to order direct, call us at 1-214-727-9217. We'd like to hear from you.

Santa Rita Software 1000 E. 14th Street, Suite 365 Plano, Texas 75074

The RESIDENT EXPERT System

Resident Expert is a trademark of The Santa Rita Company. Borland, Turbo C, Turbo Pascal, and Turbo Prolog are trademarks of Borland International Inc. IBM and PC-DOS are trademarks of International Business Machines Corporation. Lattice C is a trademark of Lattice Inc. LetsC is a trademark of Mark Williams Company. Microsoft and MS-DOS are trademarks of Microsoft Corporation.

Texas Instruments has system developers need.



"Personal Consultant™ Plus...offers a very fine expert system development and delivery tool that already has a proven record with end-users."

- Susan Shepard, AI Expert

Personal Consultant Plus 3.0 Standard Features

- Frames, rules, meta rules and procedures
- Forward/backward chaining
- Confidence factors
 Regression testing and rule tracing
- Requestion testing and role tracing
 End-user explanation facilities
 Graphics image capture and display
 Interfaces to dBase ", Lotus 1-2-3", DOS files,
 EXE or. COM programs, "C"
 Complete LISP development environment
- 2-megabyte expanded/extended memory support

- Mouse support
 Context sensitive help
 "Getting Started" tutorial-style manual

Personal Consultant Images

Optional add-on package to PC Plus (3.0)

Allows integration of "active images" into

what serious expert Power tools.



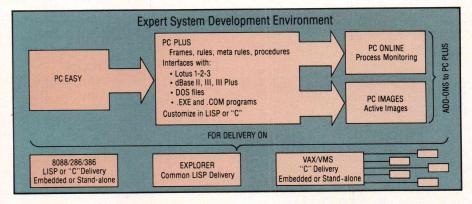
Among all the expert system development tools available for personal computers today, none deliver the power and flexibility of TI's Personal Consultant series.

Personal Consultant Easy is ideal for getting started, and is upwardly compatible with the higher functionality of PC Plus. For experienced developers, Personal Consultant Plus and its optional add-on enhancements, Online and Images, were designed to help solve a broader range of complex problems.

package helps deliver expertise that is 'online all the time."

Application delivery as flexible as the tools themselves.

Delivery can be in LISP for flexibility, or "C"* for maximum speed and portability. Our "C" options support either stand-alone or "embedded" knowledge bases. Options are available for DOSbased PCs, TI's Explorer, and DEC's VAX[™] line of multi-user minis running under VMS™.



Personal Consultant Plus. Full power for an affordable price.

At \$2,950, PC Plus has proven to be one of the richest and most flexible problem-solving tools available for the development of complex knowledgebased systems. Designed to take advantage of today's more powerful 286/386 DOS-based computers, or TI's Explorer[™] Symbolic Processing System, the new 3.0 version of PC Plus provides powerful standard features and a continuing growth path with the addition of either PC Images or PC Online, or both.

Personal Consultant Images. Picture an expert system with interactive graphics.

At \$495, PC Images enables developers to create knowledge-based applications that incorporate complex graphical "active images." User-interactive dials, gauges, forms and selection images provide a more exciting visual data input and output style.

Personal Consultant Online. The expert system as part of the process. At \$995, PC Online allows the devel-

"Texas Instruments has done more than any other company to educate people about AI, to popularize it, and to make useful AI tools available at reasonable prices."

- Jim Seymour, PC Magazine.

Technical support, training courses and Knowledge Engineering Services are available for the Personal Consultant products. If you have a question about any of our expert system power tools, we have the answer.

Pick up the phone and gain a powerful advantage.

Call 1-800-527-3500 for technical overviews of our products and a PC Plus case histories brochure which details how our power tools are being put to work today.

36106 © 1987 TI Personal Consultant and Explorer are trademarks of resonal Consultant and Explorer are trademarks of Texas Instruments Incorporated.
dBase is a trademark of Ashton-Tare.
Lotus 1-2-3 is a trademark of Lotus Development Corp.
VAX and VMS are trademarks of Digital Equipment Corporation.
*Available 4Q 1987.

knowledge bases Interactive dials, gauges, forms and selection

images Multiple images can be combined on same screen
- "Getting Started" tutorial-style manual

Personal Consultant Online

Optional add-on package for PC Plus (3.0)
 ONLINE expert systems that interact directly with

process data

- Multiple interfaces to data acquisition and

oper to design expert systems which analysis programs

- Knowledge base synchronization with process interact directly with process data, as opposed to input from a human oper-- Functions for historical and predicted trends
- Special user interface/reporting capabilities
- "Getting Started" tutorial-style manual ator. Designed for intelligent process monitoring applications, this optional

How Many Ways Can You Draw a Circle?

by James F. Blinn

A comprehensive roundup of circle-drawing algorithms

like to collect things. When I was young I collected stamps; now I collect empty margarine tubs—and algorithms for drawing circles. Because this is an algorithms issue, I will restrain myself from talking about stamps or margarine tubs; instead I'll show your process.

tubs; instead, I'll show you my album of circle-drawing algorithms.

It's traditional at this point in any discussion of geometry to drag in the ancient Greeks and mention how they considered the circle to be the most perfect shape. Even though a circle is such an apparently simple shape, it is interesting to see how many essentially different algorithms you can find for drawing the Greek's favorite curve.

I will be brief about some pretty obvious techniques to leave space to play with the more interesting and subtle techniques. Note that many of these algorithms might be ridiculously inefficient but are included to pad out the article. (OK, OK—they're actually included for

James F. Blinn, 195 South Wilson, Apt. 8, Pasadena, CA 91106. Dr. Blinn has been actively involved in computer graphics for the past 19 years. Since 1978 he has been at the Jet Propulsion Laboratory of the California Institute of Technology, producing animations depicting various space missions. He also produced computer graphics effects for the PBS series "COSMOS." He currently teaches courses in computer graphics at Cal Tech and at the Art Center College of Design in Pasadena.

This is a revised version of an article published in the August 1987 issue of IEEE Computer Graphics and Applications. © 1987 IEEE.

completeness.)

I'm not sure where I first heard of some of these. I will cite inventors where known, but let me just thank the world at large in case I've missed anybody.

A word about the pro-

gramming language I use—I am not using any formal algorithm display language here. These algorithms are meant to be read by human beings, not computers, so the language I present is a mishmash of several programming constructs that I'm sure will be perfectly clear to you.

The collection can be categorized by the two types of output—line endpoints and pixel coordinates. This comes from the general dichotomy of curve representation—parametric vs. algebraic.

Line Drawings

First, I'll look at line output. All the algorithms in this section operate in floating point and generate a series of *x*,*y* points on a unit-radius circle centered at the origin. You then play connect the dots.

1. Trigonometry

Evaluate sin and cos at equally spaced angles:

MOVE(1,0)

FOR DEGREES=1 to 360

RADIANS=DEGREES*2*3.14159/360.

DRAW(COS(RADIANS),SIN(RADIANS))

This has to evaluate the two trig functions at each loop, ick.



2. Polynomial Approximation

You can get a fair approximation of a circle by evaluating simple polynomial approximations to sin and cos. The first ones that come to mind are the Taylor's series:

$$\cos a \approx 1 - (\frac{1}{2})a^2 + (\frac{1}{24})a^4$$

$$\sin a \approx a - (\frac{1}{6})a^3 + (\frac{1}{120})a^5$$

These require fairly high-order terms to get very close.

A better approach is to fit lower-order polynomials to the desired endpoints and end slopes. This is effectively what happens with various commonly used Bezier curves—for example, the four control points (1,0), (1,0.552), (0.552,1), (0,1) describe a good approximation to the upper-right quarter of a circle. You can get the other three quadrants by rotating the control points.

When transformed to polynomial form, the first quadrant is:

$$\chi(t) = 1 - 1.344t^2 + .344t^3$$

$$\chi(t) = 1.656t - .312t^2 - .344t^3$$

with the parameter t going from 0 to 1.

MOVE(1,0)
FOR
$$T = 0$$
 TO 1 BY .01
 $X = 1 + T^*T^*(-1.344 + T^*.344)$
 $Y = T^*(1.656 - T^*(.312 + T^*.344))$
DRAW(X,Y)

This makes a pretty good circle—the maximum radius error is about 0.0004 at t=0.2 and t=0.8.

3. Forward Differences

Polynomials can be evaluated quickly by the technique

known as forward differences. Briefly, for the polynomial:

$$f(t) = f_0 + f_1 t + f_2 t^2 + f_3 t^3$$

if you start at t=0 and increment by equal steps of size δ , the forward differences are:

$$\Delta f = f_1 \delta + f_2 \delta^2 + f_3 \delta^3$$
$$\Delta \Delta f = 2f_2 \delta^2 + 6f_3 \delta^3$$
$$\Delta \Delta \Delta f = 6f_3 \delta^3$$

Then, for polynomials stepping in units of 0.01:

X=1; DX=-.000134056; DDX=-.000266736; DDDX= .000002064 Y=0; DY= .016528456; DDY=-.000064464; DDDY=-.000002064 MOVE(X,Y) FOR I=1 TO 100

X = X + DX; DX = DX + DDX; DDX = DDX + DDDX Y = Y + DY; DY = DY + DDY; DDY = DDY + DDDYDRAW(X,Y)

Trust me, I'm a doctor. If you don't believe it, look up forward differences on page 328 of Newman and Sproull—I'm not going to do all the work here.

Notice the number of significant digits in the constants. It might seem as though that many digits would require double precision, but in practice the accumulated round-off error using single precision is less than the error due to the polynomial approximation.

4. Incremental Rotation

Let's back off from the approximation route and try another approach. Start with the vector (1,0) and multiply it by a one-degree rotation matrix each time through the loop:



DRAWING CIRCLES (continued from page 19)

 $\begin{aligned} \text{DELTA} &= 2*3.14159/360. \\ \text{SINA} &= \text{SIN(DELTA)} \\ \text{COSA} &= \text{COS(DELTA)} \\ X &= 1; \ Y &= 0 \\ \text{MOVE(X,Y)} \\ \text{FOR I} &= 1 \ \text{TO } 360 \\ \text{XNEW} &= \text{X*COSA} - \text{Y*SINA} \\ \text{Y} &= \text{X*SINA} + \text{Y*COSA} \\ \text{X} &= \text{XNEW} \end{aligned}$

5. Extreme Approximation

If the incremental angle is small enough, you can make the approximations $\cos a = 1$ and $\sin a = a$. The number of times through the loop is $n = 2\pi/a$, or conversely, the angle is $a = 2\pi/n$, depending on which you want to use as input:

A = .015; N = 2*3.14159/A X = 1; Y = 0 MOVE(X,Y)FOR I = 1 TO N XNEW = X - Y*A Y = X*A + Y X = XNEW X = XNEWX = XNEW

But there's a problem. Each time through the loop, you are forming the product:

$$(x_{new}, y_{new}) = (x_{old}, y_{old})$$
 $\begin{pmatrix} 1 & a \\ -a & 1 \end{pmatrix}$

The matrix is almost a rotation matrix, but its determinant equals $1+a^2$. This is bad. It means that the running (x,y) is magnified by this amount on each iteration, so what you get is a spiral that gets bigger and bigger. How to

fix this?—introduce a bug into the algorithm.

6. Unskewing the Approximation

Because vector multiplication and assignment don't occur in one statement, you had to calculate *y* carefully, using the old value for *x*. Suppose you were dumb and did it the naive way:

A = .015; N = 2*3.14159/A X = 1; Y = 0 MOVE(X,Y)FOR I = 1 TO N X = X - Y*A Y = X*A + YDRAW(X,Y)

Now, what is the effect of this? Really what you get is:

 $x_{new} = x_{old} - y_{old}a$ $y_{new} = x_{new} a + y_{old} = x_{old} a + y_{old} (1-a^2)$

In other words:

$$(\chi_{new},y_{new}) = (\chi_{old},y_{old}) \quad \left(\begin{array}{cc} 1 & a \\ -a & 1-a^2 \end{array} \right)$$

This matrix has a determinant of 1, and there is no net spiraling effect. What you get is actually an ellipse that is stretched slightly in the northeast-southwest direction and squeezed slightly in the northwest-southeast direction. The maximum radius error in these directions is approximately a/4.

Now comes the really interesting part. Because you can start out with any vector, let's try (1000,0). Now, cleverly select a to be an inverse power of 2 and the multiplication becomes just a shift—for example, a value of a=1/64 is just (shift right 6) and generates the circle in about 402 steps. So, you can do all this just with integer arithmetic and no multiplication. This is how you used to draw circles quickly—and in fact do rotation incrementally—before the age of hardware floating point and even hardware multiplication. (This was probably invented by Ivan Sutherland.)

7. Rational Polynomials

Another polynomial tack can be taken by looking in your hat and pulling out the following rabbit:

if:

 $x = \frac{(1-t^2)}{(1+t^2)}$ $y = \frac{2t}{(1+t^2)}$

then:

 $x^2 + y^2 = 1$

no matter what t is (or identically, as mathematicians would say). Running t from 0 to 1 gives the upper-right quadrant of the circle. You can again evaluate these poly-

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features.

Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF

into your own custom editor containing the commands and features you desire. It's fast and easy.

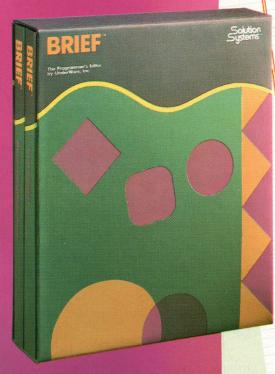
Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "Recommended. If you need a general purpose PC programming editor, look no further." His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. BRIEF 2.0 is just \$195. If you already own BRIEF, call for upgrade information.

TO ORDER CALL: 1-800-821-2492 (in MA call 617-337-6963)

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

So. Weymouth, MA 02190 (617) 337-6963



Requires an IBM PC or compatible with at least 192K RAM. BRIEF is a trademark of UnderWare, Inc. Solution Systems is a trademark of Solution Systems

Look at these BRIEF 2.0 enhancements!

- All new documentation with tutorials on basic editing, regular
- expressions and the BRIEF Macro Language.
- Setup program for easy installation and configuration.

 (Peguires no knowledge of the macro language) (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and
- Expanded regular expressions, with matching over line
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete
- characters, specify command parameters).
- Support for more programming languages.
- Enhanced large display support, including wider displays. Optional borderless windows.
- Reconfigurable indenting for C files

(supports most indenting styles).

Plus the basic features that made features **SO** popular! BRIEF **SO** popular!

Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space Automatic language sensitive indentation

CIRCLE 142 ON READER SERVICE CARD

DRAWING CIRCLES

(continued from page 20)

nomials by forward differences, stepping *t* in increments of 0.01, and get:

X=1; DX=-.0001; DDX=-.0002 Y=0; DY= .02 W=1; DW= .0001; DDW= .0002 MOVE(X,Y) FOR I=1 TO 100 X=X+DX; DX=DX+DDX Y=Y+DY W=W+DW; DW=DW+DDW DRAW(X/W,Y/W)

Note that this is not an approximation like the last few tries were. It is exact—except for round-off error. Even round-off error can be removed, either by calculating the polynomials directly or by scaling all numbers by 10,000 and doing it with integers. (The division x/w must still be done in floating point.)

This one has always amazed me—effectively, you get to evaluate two transcendental functions exactly with only a few additions. What's the catch? It's an application of the No Free Lunch theorem—you don't get to pick the angles. If you watch the points, you see that they are not equally spaced around the circle. In fact, as t goes to infinity, the point keeps going counterclockwise but slows down, finally running out of juice at (-1,0). If you go backward to minus infinity, the point goes clockwise, finally stopping again at (-1,0). (Yet more evidence that $-\infty = +\infty$.) To draw a complete circle, you are best advised to run t from -1 to +1, which draws the whole right half, and then mirror it to get the left half.

8. Differential Equations

An entirely different technique is to describe the motion of (x,y) dynamically. Imagine the point rotating about the center as a function of time t. The position, velocity, and acceleration of the point will be:

$$(x,y) = (\cos t, \sin t)$$

 $(x', y') = (-\sin t, \cos t) = (-y,\chi)$
 $(x'', y'') = (-\cos t, -\sin t) = (-\chi, -y)$

You can cast these into differential equations and use several numerical integration techniques to solve them.

The dumbest one, Euler integration, is just:

$$\begin{aligned} x_{new} &= x_{old} + x'_{old} \, \Delta t = x_{old} - y_{old} \, \Delta t \\ y_{new} &= y_{old} + y'_{old} \, \Delta t = y_{old} + x_{old} \, \Delta t \end{aligned}$$

This looks a lot like algorithm 5 and has the same spiraling-out problem. You can generate better circles by using better integration techniques. My two favorites are the "leapfrog" technique and the Runge-Kutta technique.

Leapfrog calculates the position and acceleration at times:

$$t, t + \Delta t, t + 2\Delta t, \ldots$$

but calculates the velocity at times halfway between them:

$$t+\frac{1}{2}\Delta t, t+\frac{3}{2}\Delta t, \ldots$$

Advancing time one step then looks similar to Euler with just the evaluation times offset:

$$x_{t+\Delta t} = x_t + x'_{t+\frac{1}{2}\Delta t} \Delta t$$

$$\chi'_{t+\frac{3}{2}\Delta t} = \chi'_{t+\frac{1}{2}\Delta t} + \chi''_{t+\Delta t}\Delta t$$

(with similar equations for y). The position and velocity leapfrog over each other on even/odd half-time steps, so you have to keep separate variables for the velocities χ' and y' The code has a lot in common with algorithm 6 and probably for good reason:

$$\begin{split} X = 1 \ ; \ Y = 0 \\ VX = -SIN(DT/2); \ VY = COS(DT/2) \\ MOVE(X,Y) \\ FOR \ I = 1 \ TO \ N \\ X = X + VX*DT \qquad "update \ posn" \\ Y = Y + VY*DT \\ VX = VX - X*DT \qquad "update \ veloc, \ AX = -X" \\ VY = VY - Y*DT \qquad " \qquad AY = -Y" \\ DRAW(X,Y) \end{split}$$

Runge-Kutta is a slightly involved process that takes a fractional Euler step, reevaluates the derivatives there, applies the derivative at the original point, steps off in this new direction, generally screws around, and finally takes some average of all these to get the new time step. Plugging the differential equation into the formulas and simplifying requires about a page of algebra. You can look up the actual equations; they're not incredibly complicated, but their derivation is "beyond the scope" of almost all numerical analysis textbooks I have seen.

One advantage of Runge-Kutta is that it finds the position and velocity at the same time step, so for circles you can generate *x* and *y* with the same computation. Another advantage is that it comes in second-order, third-order, fourth-order, and so on versions for higher orders of precision than does leapfrog.

Plugging in for second-order RK, the ultimate result is:

$$x_{new} = x_{old}(1 - \frac{1}{2}\Delta t^2) + y_{old}(-\Delta t)$$

$$y_{new} = x_{old}(\Delta t) + y_{old}(1 - \frac{1}{2}\Delta t^2)$$

Does this look familiar? The third-order RK and another page of algebra leads to:

$$\begin{aligned} x_{new} &= x_{old} (1 - \frac{1}{2} \Delta t^2) + y_{old} (-\Delta t + \frac{1}{6} \Delta t^3) \\ y_{new} &= x_{old} (\Delta t - \frac{1}{6} \Delta t^3) + y_{old} (1 - \frac{1}{2} \Delta t^2) \end{aligned}$$

Guess what fourth-order RK gives . . . you're right. I won't even bore you with the code.

··· a real UNIX System V for your 386 or 286-based PC!"

Microport System V/AT and Microport System V/386 are fully-licensed implementations of the certified AT&T UNIX System V Release 3 for personal computers.

Microport System V provides true multiuser and multi-tasking capabilities. It runs in protected mode and supports the entire address space available; up to sixteen megabytes on an AT and up to four gigabytes on an 80386. It supports all standard hard disks, multiple printers, multiple modems, 2 to 17 users on a single AT, and an unlimited number of users on an 80386.

Additional features:

- Demand-paged, virtual memory management.
- Virtual consoles allow fifteen virtual windows of operation for UNIX applications and DOS windows under DOS Merge.
- Electronic mail, communications with uucp and CU.
- Dynamic buffer allocation.
- PC-DOS partitioning allows DOS and System V files to reside on the same hard disk (not required with DOS Merge).
- Shared libraries.
- 80286 compatibility.
- Link kit to allow user installable device
- Menu-driven system administration.
- On-line help facility.

Microport Sys Vision™ is a part of both System V/AT and System V/386, so you won't have to worry about system administration. Both are menu-driven and have context-sensitive help facilities. Sys Vision menus lets you perform all the system housekeeping and basic tasks without knowing the specific UNIX commands. It is menu-driven and has context-sensitive help facilities.

RUNTIME PACKAGE. The System V Runtime Package contains over 180 utility programs. It is one of the most comprehensive UNIX runtime systems currently available and includes the screen editor "vi" with user tutorials.

System V is sold with a two-user UNIX binary license. For an additional fee, users can obtain a kit to upgrade this license to an unlimited number of users.

Ours \$ 169 List \$ 199 80286: List \$ 199 Ours \$ 169 80386:

UNLIMITED USER LICENSE KIT.

Upgrades your System V Runtime for an unlimited number of users.

List \$ 249 Ours \$ 209 80286: 80386: List \$ 249 Ours \$ 209

SOFTWARE DEVELOPMENT

SYSTEM. This programming package includes an AT&T C Compiler that makes full and efficient use of your processor's extended instruction set. The code produced is among the most compact and fastest available.

Sdb, the source level debugger, allows display of the actual lines of C code being executed. Many useful tools such as SCCS (source code control system), make and ctrace are also provided.

80286: List \$ 249 Ours \$ 209 List \$ 499. Ours \$ 429 80386:

TEXT PROCESSING SYSTEM. This package consists of the complete System V Release 2 Documentors Workbench (DWB). It includes both the new troff (device-independent troff) and the old troff (otroff). Drivers for the HP LaserJet Printer and APPLE's LaserWriter are optionally available.

Ours \$ 169 80286: List \$ 199 List \$ 199 Ours \$ 169 80386:

COMPLETE SYSTEM V. Includes Runtime Package, Software Development Package and Text Processing Package.

List \$ 549 Ours \$ 465 80286: List \$ 799 Ours \$ 699 80386:

DOS Merge. DOS Merge is a true multiuser and multi-tasking environment that supports concurrent use of both UNIX and DOS. With DOS Merge, you can develop software by combining DOS and UNIX programs and commands. You can even run both UNIX and DOS programs that use the same files. And DOS Merge 386 uses the full capabilities of the 80386's memory management so you can run multiple DOS programs and UNIX programsatthesametimeondumbterminals. DOS Merge 386 comes in two versions: a 2-user version and an unlimited-user version.

List \$ 149 Ours \$ 125 80286: 80386 Unltd: List \$ 495 Ours \$ 429 80386 2 User: List \$ 395 Ours \$ 345

CIRCLE 98 ON READER SERVICE CARD

Now Available From Programmer's Connection

Microport Products

List Ours

System V/386 Complete System	799	699
Runtime System (2 Users)	199	169
Software Development System	499	429
Text Preparation System	199	169
System V/ 386 Unlimited User License	249	209
DOSMerge 386 2-user System	395	345
DOSMerge 386 Unlimited-user System	495	429
	F 40	AOF
System V/ AT Complete System	549	465
System V/ AT Complete System		465 169
Runtime System (2 Users)	199	
Runtime System (2 Users)	199 249	169
Runtime System (2 Users)	199 249 199	169 209
Runtime System (2 Users) Software Development System Text Preparation System System V/ AT Unlimited User License	199 249 199 249	169 209 169 209
Runtime System (2 Users)	199 249 199 249	169 209 169 209

Prices include FREE UPS surface shipping to all U.S. customers. Express services are available at the shipping carrier's standard rate.

When ordering, please specify 80286 or 80386 computer. DOSMerge 286 doesn't work on all clones, please ask before ordering.

Please refer to our main advertisement in this journal for more information and the largest advertised selection of programmer's development tools for IBM personal computers and compatibles.

> Programmer's Connection Hartville, Ohio 44632

Hours: Weekdays 8:30 AM to 8:00 PM EST.

CALL TOLL FREE

USA 800-3	36-1166
CANADA 800-2	25-1166
OHIO & ALASKA (Collect) 216-8	77-3781
	77-3781

CUSTOMER SERVICE 216-877-1110 9102406879 TELEX 62806530 **EASYLINK**



DRAWING CIRCLES

(continued from page 22)

9. Half Interval

The half-interval method (suggested by Jim Kajiya) assumes you have two endpoints of an arc and wish to fill in the middle with points on the circle. At each step you insert a new point between two others. Assuming a circle centered at the origin, the new point will be approximately halfway between the surrounding ones:

$$(x_{my} y_m) = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$$

It just needs to be moved outward to lie on the circle, which involves scaling the previous expression to length 1. If the original points are at unit distance from the origin, this means dividing by $\sqrt{1 + x_1x_2 + y_1y_2}/\sqrt{2}$.

By doing this recursively, you can keep splitting until some error tolerance is met. The code is something like:

X1=1; Y1=0 X2=0; Y2=1 MOVE(X1,Y1) SPLIT(X1,Y1, X2,Y2)

where SPLIT(X1,Y1, X2,Y2) is defined to be:

D = SQRT(2*(1+X1*X2+Y1*Y2))
XM = (X1+X2)/D
YM = (Y1+Y2)/D
IF error_tolerance_ok
DRAW(XM,YM)
DRAW(X1,Y2)
ELSE
SPLIT(X1,Y1, XM,YM)
SPLIT(XM,YM, X2,Y2)

The error tolerance could be just a recursion depth counter, stopping at a fixed recursion depth. This is nice because, for a given pair of initial points, the value of *D* is just a function of recursion depth and can be precomputed and placed in a table.

Pixel-Based Techniques

The other major category of algorithms involves output more directly suited to raster displays. Here the question is not where to move the "pen" next but which of the grid of pixels to light up. The preceding algorithms can, of course, be applied to pixels by generating coordinates and feeding them to a line-to-pixel drawing routine, but I won't pursue these methods. I'll just look at ways to generate the desired pixels directly. For simplicity I will assume you are drawing a 100-pixel-radius circle with pixels addressed so that (0,0) is in the center and that negative coordinates are OK. The algorithms operate in integer pixel space, assuming square pixels. Note that the following variables start with *I*, indicating that they are integers.

10. Fill Disk

Perhaps the dumbest algorithm is just to see how far each

pixel is from the center and color it in if it's inside the circle:

FOR IY = -100 TO 100 FOR IX = -100 TO 100 IF (IX*IX + IY*IY < 10000) SETPXL(IX,IY)

This, of course, fills in the entire disk instead of just drawing lines, but who's being picky? You would be correct in assuming that this might be a bit slow. Some quick speedups: calculate the value of χ^2 by forward differences and calculate the allowable range of χ^2 outside the χ loop (forward differences probably aren't worth the trouble for this).

FOR IY = -100 TO 100 IX2MAX = $10000 - IY^*IY$ IX2 = 10000; IDX2 = -199; IDDX2 = 2 FOR IX = -100 TO 100 IF (IX2 < IX2MAX) SETPXL(IX,IY) IX2=IX2+IDX2; IDX2=IDX2+IDDX2

11. Solve for X Range Covered

The preceding algorithm still examines every pixel on the screen. You can skip some of this by explicitly solving for the range in x:

FOR IY = 100 TO -100 BY -1IXRNG = SQRT(10000 $-IY^*IY$) FOR IX = -IXRNG TO IXRNG SETPXL(IX,IY)

Or just plot the endpoints instead of filling in the whole disk:

FOR IY = 100 TO -100 BY -1IX = SQRT(10000-IY*IY) SETPXL(-IX,IY) SETPXL(IX,IY)

This leaves unsightly gaps near the top and bottom.

12. Various Approximations to SQRT

Make a polynomial, or rational polynomial, approximation to $\sqrt{10000-y^2}$ that is good for the range -100...+100. Evaluate it with forward differences.

13. Driving X Away

Let's just do the upper-right quarter of the circle and follow the point (0,100). For each downward step in y, you move to the right some distance in x. Start at the x that's left over from last time and step it to the right until it hits the circle, leaving a trail of pixels behind:

$$\begin{split} \text{IX=0} \\ \text{FOR IY=100 TO 0 BY -1} \\ \text{IX2MAX = 10000-IY*IY} \\ \text{DO UNTIL (IX*IX)>IX2MAX} \\ \text{SETPXL(IX,IY)} \\ \text{IX=IX+1} \end{split}$$

Calculation of IX2MAX and IX2 can be done by forward

differences:

```
IX = 0

IX2=0; IDX2=1; IDDX2=2

IX2MAX=0; IDX2MAX=199; IDDX2MAX=-2

FOR IY = 100 TO 0 BY -1

DO UNTIL IX2 > IX2MAX

SETPXL(IX,IY)

IX=IX+1

IX2=IX2+IDX2; IDX2=IDX2+IDDX2

I X2MAX=I X2MAX+ID X2MAX

IDX2MAX=IDX2MAX+IDDX2MAX
```

This still has a few problems, but I won't pursue them because the next two algorithms are so much better.

14. Bresenham

The previous algorithm begins to look like Bresenham's algorithm, which is the top of the line in pixel-oriented circle algorithms. It endeavors to generate the best possible placement of pixels describing the circle with the smallest amount of (integer) code in the inner loop. It operates with two basic concepts.

First, the curve is defined by an "error" function. For my circle, this is $E=10000-\chi^2-y^2$. For points exactly on the circle, E=0; inside the circle, E>0; and outside the circle, E<0.

Second, the current point is nudged by one pixel in a direction that moves "forward" and in a direction that minimizes E. Consider just the octant of the circle from (0,100), moving to the right by 45 degrees. At each iteration, you can choose to move either to the right (R)—x=x+1—or diagonally (D)—x=x+1 and y=y-1.

The nice thing about this is that the value of E can be tracked incrementally. If the error at the current (x,y) is:

$$E_{cur} = 10000 - x^2 - y^2$$

then an R step will make:

$$E_{new} = 10000 - (\chi + 1)^2 - y^2$$

= $E_{cur} - (2\chi + 1)$

and a D step will make:

$$E_{new} = 10000 - (x + 1)^2 - (y-1)^2$$

= $E_{cur} - (2x + 1) + (2y-1)$

ED	ER	ED - ER
+	+	$E_D - E_R = 2y-1$, always positive
+	-	$ E_D + E_R $
_	+	can never happen
_	-	$-E_D + E_R = -(2y-1)$, always negative

Table 1: Testing the sign of $|E_D| - |E_R|$.

Now, for the octant in question, $x \le y$, $x \ge 0$, and y > 0. So, an R step subtracts something from E and a D step adds something to E. The naive version of the algorithm determines which way to go by looking at the current sign of E, always striving to drive it toward its opposite sign:

$$IX=0; IY=-100$$

$$IE=0$$

$$WHILE IX <= IY$$

$$IF (IE < 0)$$

$$IE=IE+IY+IY-1$$

$$IY=IY-1$$

$$IE=IE-IX-IX-1$$

$$IX=IX+1$$

$$SETPXL(IX,IY)$$

15. Improved Bresenham

You can do better. What you want to do at each step is actually to pick the direction that generates the smallest-size error, |E|. You want to look ahead at the two possible new error values:

$$E_R = E - (2\chi + 1)$$

 $E_D = E - (2\chi + 1) + (2\gamma - 1)$

and test the sign of $|E_D| - |E_R|$. The trick is to avoid calculating absolute values. Table 1, below, shows the possibilities.

Now comes the tricky part. You can define a "biased" error from the (+-) case:

$$G = E_D + E_R$$

and use this as the test for all three cases. This works for the following reason: In the (++) case, $|E_D|-|E_R|=2y-1$ is positive, but so is G. In the (--) case, $|E_D|-|E_R|=-(2y-1)$ is negative, but so is G.

G can be calculated incrementally, just like E was. The new values due to R and D steps are:

$$G_R = G-4x-6$$

 $G_D = G-4x+4y-10$

Further, the increments to G can be calculated incrementally. You get the idea by now . . .

```
IR = 100
IX = 0; IY = IR
IG = 2*IR-3
IDGR = -6; IDGD = 4*IR-10
WHILE IX<=IY
        IF IG<0
                               "go diagonally"
           IG = IG + IDGD
           IDGD = IDGD - 8
           IY = IY - 1
        FLSE
           IG = IG + IDGR
                               "go right"
           IDGD=IDGD-4
        IDGR = IDGR - 4
         IX = IX + 1
        SETPXL(IX,IY)
```

DRAWING CIRCLES (continued from page 25)

Whew!

Conclusion

So, why is all this interesting—aside from the pack rat joy of collecting things?

Well, you can certainly use these algorithms to optimize your circle-drawing programs if you're into circles. Each algorithm has its own little niche in the speed-accuracy-complexity trade-off space. Sometimes economy is misleading—the *SETPXL* routine often gobbles up any time you saved being clever with Bresenham's algorithm. Let's face it: unless there's something very time-critical, I usually use algorithm 1 because it's easiest to remember.

The really interesting thing about all these algorithms is the directions in which they lead you when you try to generalize them. Algorithms 2 and 7 lead to general polynomial curves. Algorithm 4 leads to iterated function theory. Algorithm 5 leads to the CORDIC method of function evaluation. Algorithm 11 has to do with rendering spheres. (I wonder what happens to algorithm 15 if you use some other simple functions of x and y for G, G_B, and G_D.) In fact, although many of these algorithms look quite similar when applied to circles, their generalizations lead

to very different things. It sort of shows the underlying unity of the universe. Maybe the Greeks had something there.

Bibliography

Newman, William M.; and Sproull, Robert F. *Principles of Interactive Computer Graphics*. New York: McGraw-Hill, 1979.

Turkowski, Kenneth. "Anti-Aliasing Through the Use of Coordinate Transformations." ACM Transactions on Graphics, vol. 1, no. 3 (July 1982): 215–234. Refers to CORDIC.

For references to Runge-Kutta, see any numerical analysis text—for example, *Schaum's Outlines*.

DD.

Vote for your favorite feature/article. Circle Reader Service No. 1.

Structured Analysis breakthrough!

THE FIRST **COMPLETE**SA SOFTWARE FOR UNDER \$1,000. Discover the power of

computer-aided Structured Analysis...Create specifications more efficiently, more accurately,...With **Teamwork/PCSA**, a complete, automated SA tool for your PC **for only \$995.**

No other system includes these features for under \$1,000:

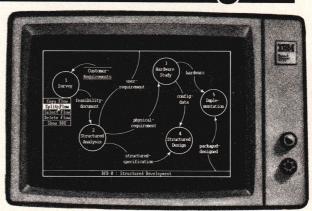
- Full support of Yourdon/DeMarco SA techniques
- Easy-to-use mouse driven interface with pop-up menus
- Includes integrated project data dictionary
- Includes consistency checking and diagram balancing
- Now also includes P-Specs, Postscript™ output, and new, easy tutorial

Teamwork/PCSA runs on most IBM®—compatible PCs. It's used by leading developers at Boeing, AT&T, GE, HP, and Bank of America. And it's the only PC-based software that offers you a growth path to the Teamwork family of CASE tools for real-time modeling, system design and lifecycle management.



Cadre Technologies Inc. 222 Richmond St. Providence, RI 02903

IBM is a registered trademark of International Business Machines. Postscript is a registered trademark of Adobe Systems. Inc.



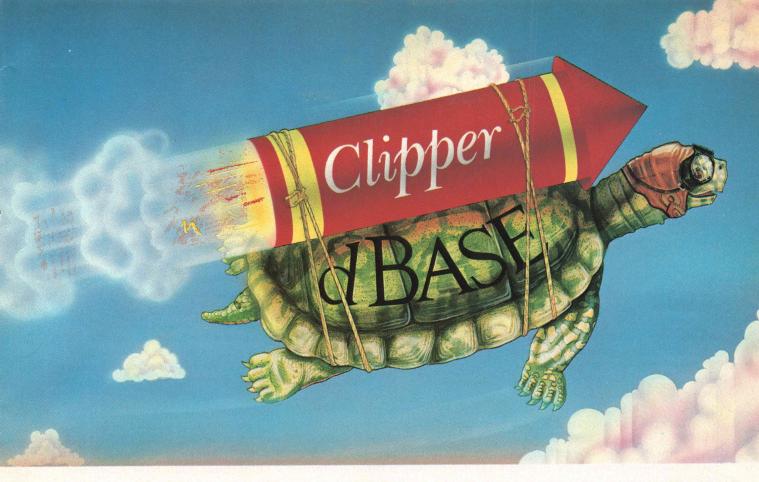
FREE DeMarco Book! We'll give you

two reasons to order **Teamwork/PCSA** today. ON E: you get a 30-day money-back guarantee, so there's absolutely no risk. TWO: Order now and we'll send you **Structured Analysis and System Specification** by Tom DeMarco. It's the "Bible" of structured analysis and normally sells for over \$40. And it's yours free. For details or to place your order, call or write today.

CALL (401) 351-CASE

North American prices only. Volume discounts available

CIRCLE 261 ON READER SERVICE CARD



Turtle Soupe

Real programmers don't use dBASE. Or do they?

We're finding that some very swift programmers are using it to

write some very fast applications, and are completing their projects much more quickly.

But they cheat.

They use our Clipper™ compiler to combine dBASE™ with C and assembler.

With dBASE used like pseudo-code, they can then quickly create

prototypes that actually run.

Then, with dBAŚĒ doing the high-level database functions, they use our Clipper compiler to link in C or assembly language modules from their own bag of tricks.

And they're finding that they're linking in less than they expected because Clipper compiled code runs so fast and because of Clipper's built-in enhancements.

Clipper includes easy networking that provides file and record locking the way it should be done.

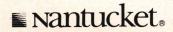
Fast screens that can be treated as memory variables and eliminate the need for direct screen writes and all that tortuous heap management code.

Box commands that make windowing a breeze. And more.

So if you'd like to use your time more productively, check us out: Nantucket Corporation, 12555 W. Jefferson Boulevard, Los Angeles, CA 90066.

Or if you're on deadline, call (213) 390-7923 today.

Clipper could get you out of the soup.



File Comparison Algorithms

by Tom Steppe

everal popular algorithms exist for comparing two files. All of these actually look first for matches rather than differences. After the matching process has been completed, the remainders of the files that are not included in the matches are then reported as differences. (See Figure 1, page 29.)

The algorithms differ greatly in their conceptualization of the problem, however. In this article, I examine several algorithms for comparing text files—specifically, source code files—using a line as the basic unit of comparison. The ideas and algorithms I present here, however, can be extended to other types of files and other units of comparison as well. I also present a new algorithm with some interesting properties.

Evaluating The Algorithms

Any file comparison algorithm should be evaluated according to several criteria:

- Is it efficient? Time efficiency (speed) and space efficiency (memory usage) are both practical considerations. Usually they are related to the lengths of the files being compared.
- Is it robust? No algorithm is flawless. For any given file comparison algorithm, it is always possible to concoct devious situations in which its performance appears less than perfect. The algorithm should, however, be able to produce reasonable difference reports for a variety of test cases.
- Can it let differences go undetected? No algorithm should allow a file difference to go undetected.

Tom Steppe, P.O. Box 2887, Ann Arbor, MI 48106. Tom designs and develops software written exclusively in C.

Determining which files are more equal than others

• Can it let matches go undetected? If an algorithm can overlook matching lines, it will report these lines as differences when they are not. If the file comparison is being performed to produce a delta file, this usually is not a major problem, even though each undetected match does increase the size of the delta file unnecessarily. If the differences are to be inspected visually, however, a report of false differences can be a serious drawback.

Say, for example, that you do not have a file comparison utility and so you have to compare two files by eye. This process is certainly tedious and prone to error, especially if some of the differences are subtle. If you now use a file comparison utility that is known to report false differences, you have to inspect the output by eye and decide which reported differences are true differences. The utility has not really done the job for you, it has only made your "by eye" inspection a smaller job that is still prone to error.

• Can it detect blocks of text that have been moved? Typically, if a block of text has been moved, it simply shows up in the report of differences as a large deletion of text at one location and a large insertion of text at another. Unfortunately, no differences within the moved block are highlighted.

When a file comparison is used to create a delta file, the ability to detect

moved blocks of text is probably desirable because it can lead to smaller delta files. But, when a file comparison is performed so that the differences can be inspected visually, the ability to detect moved blocks is not always as handy as it might seem to be. Trying to report the moved blocks is often difficult and can lead to complicated reports of the differences, especially when a large block of text is moved, a piece of that block is moved to another location, a piece of that piece is moved to still another location, and so on. Also, the difference report can sometimes be overburdened by uninteresting reports of small blocks (one-line and two-line blocks of text) being moved all over the place.

Only one algorithm discussed here can inherently detect moved blocks of text. The other algorithms, however, can be extended to do so, as follows. After applying the algorithm, replace each matching line in each file with a line that is guaranteed never to match. This leaves only the differences, which could contain moved blocks of text. Next, reapply the algorithm to the transformed files. Any match that is found in this pass will represent a moved block of text (see Figure 2, page 29). Continue this process iteratively until no new matches can be found. Of course, the cost of this iterative behavior is longer execution time.

These criteria help to provide a useful basis for surveying popular file comparison algorithms.

Popular Algorithms for Finding Matches

Scan Until Next Match

The "scan until next matching se-

quence" algorithm is probably the oldest method of file comparison. This algorithm starts at the tops of both files and matches as many lines as possible. When a difference is detected, the next M lines are scanned until at least N consecutive matching lines are found. If a sequence of N or more consecutive matching lines is found, the process begins again after the matching sequence. If such a sequence is not found, the process begins again M lines further down in the files. This process is repeated until the ends of the files are reached.

The values of M and N can be adjusted to affect the algorithm's performance. The value of M is used to control efficiency by restricting the number of lines that will be examined while searching for a sequence of matching lines. When an improper sequence of matching lines is discovered, the algorithm can be reapplied using a new value for N that is larger than the length of the improper sequence. In this way, the algorithm will overlook the undesirable sequence because it contains fewer than N matching lines, but as is always the case, the algorithm will also overlook any legitimate matching sequences that contain fewer than N lines (see Figure 3, page 30). Unfortunately, these matching lines are then reported as differences. All too often, this algorithm produces bad reports in common situations.

Although this algorithm is often highly time efficient, requires minimal memory, and frequently produces good difference reports, it does not take long to become frustrated with its shortcomings and inherent problems and begin looking for a better solution.

Longest Common Subsequence

Think of a file as representing a sequence of lines. A subsequence of those lines is defined simply as any sequence of lines that results from removing zero or more lines from the original sequence—for example, the longest subsequence of any sequence of lines is the sequence itself, with zero lines removed. Also, a sequence of zero lines would be a subsequence of any sequence because it could be created by removing all the lines from any sequence.

The "longest common subse-

quence" approach to file comparison takes the two files to be compared and finds the longest sequence of lines that is a subsequence of each of the files' lines—the longest common subsequence (see Figure 4, page 30). The details of the algorithm are not discussed here, but sources of such discussions are included in the bibliography. The Unix diff command is based on this algorithm.

This algorithm provides a simple, compact formalization of the file comparison problem and produces reasonable difference reports in a variety of test cases. The reports are quite acceptable whether the comparison is being used for visual inspection of the differences or for creating a delta file. In fact, among all the algorithms discussed here, it is probably safe to say that this one con-

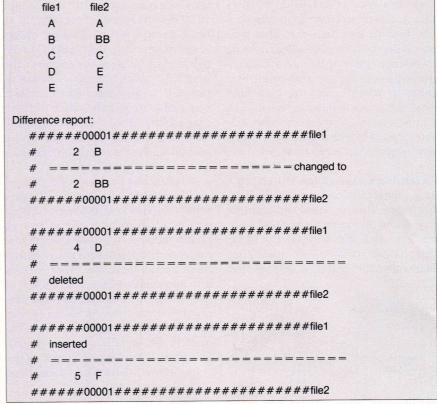


Figure 1: File comparison algorithms actually look first for line matches and then report lines that are not included in the matches as differences. The differences are usually expressed as the changes, insertions, and deletions that can be applied to one file to make it identical to the other.

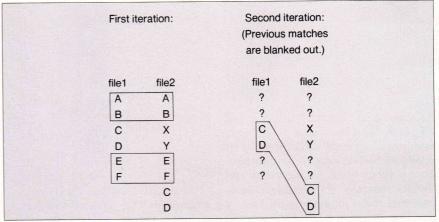


Figure 2: Moved blocks of text can be found by applying a standard linematching algorithm to the files and then reapplying the algorithm iteratively to the remainders of both files.

FILE COMPARISONS (continued from page 29)

sistently produces the best reports when comparing files that do not involve blocks of text that have been moved.

Sometimes the quality of the reports can be overshadowed by issues of time and space efficiency. This is not always true, but situations that include a poor combination of large files and limited computer resources can lead to less than desirable performance by this algorithm. A basic implementation of the algorithm requires linear space and quadratic time. In some cases, the quadratic time can prove to be unacceptable. In summary, the "longest common subsequence" algorithm produces excellent reports, but it can be slow.

Extended Unique Line Matching

The "extended unique line matching" algorithm is based on the idea that a line that occurs once and only once in each file must be the same line. These pairs of "unique" lines determine the initial set of matched

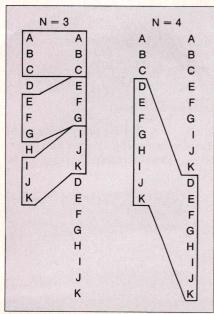


Figure 3: The "scan until next matching sequence" algorithm often produces bad reports in common situations. When N=3, the algorithm settles for matches of three lines, never realizing that a match of eight lines is possible. When N=4, it discovers the match of eight lines but does not detect the remaining match of three lines (A, B, C).

lines. (Imaginary lines at the tops and the bottoms of the files are also added to the set of matched lines.) Then, in each file, the lines adjacent to each match are examined and, if identical, are added to the set of matched lines. This process is repeated until no new matches can be found.

This algorithm has strong intuitive appeal. It is efficient, being linear in both time and space. Also, it is the only popular algorithm that inherently detects blocks of text that have been moved (even if some differences exist within the blocks). Moved blocks can be detected because the search for pairs of unique lines is in no way sequential and, therefore, can result in matches that indicate that a block of text has been moved. Note that the algorithm can find a moved block of text only if it contains a unique line match within it.

A significant problem with this algorithm is that it is prone to allowing some matches to go undetected. This occurs when matching lines are not neatly flanked by either unique line matches or the adjacent matches that have grown outward from unique line matches (see Figure 5, below).

This algorithm is fast and can frequently detect moved blocks of text, but a sacrifice is often made in the quality of the difference report. Probably its best application is in the generation of delta files when speed is the primary concern.

A New Algorithm

The "recursive longest matching sequence" algorithm uses a simple yet effective approach to the problem.

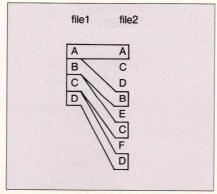


Figure 4: The 'longest common sequence' algorithm finds the longest (not necessarily consecutive) sequence of lines that is contained in both files.

This method first scans both files from beginning to end, looking for the longest sequence of consecutive matching lines. That sequence is then thought of as dividing each of the two files into an upper section and a lower section. Then, the algorithm proceeds by scanning both upper sections looking for the longest sequence of consecutive matching lines and, similarly, both lower sections for the same. These matching sequences then divide their respective sections, and the process continues recursively until no more matches can be found.

This method of file comparison is easy to understand and produces acceptable difference reports across a spectrum of test cases. It uses linear space but quadratic time. Because time efficiency can be a problem in some situations, a simple modification of the algorithm is needed. An explanation of the modification requires an understanding of the method used to locate the longest sequence of matching lines between sections of two files.

First of all, once the longest sequence is known, it can be identified by a pair of starting lines—one line from each file that specifies where the sequence begins in that file. So, when searching for the longest sequence, candidate pairs of starting lines are examined successively (in some intelligent order that starts at the beginnings of both file sections), and information is continually maintained about the length and location of the longest sequence of matching lines that has been discovered so far.

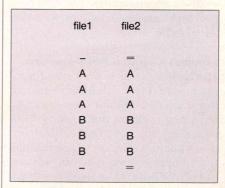


Figure 5: The "extended unique line matching" algorithm is prone to detecting false differences. In this case, no matches are found (because there are no unique line matches) and all lines are reported as differences.

Protected Mode and 32-bit Performance Today

Introducing OS/286™ and OS/386™. extensions to MS-DOS 3.x that enable full use of the 80286 and 80386. Now you get direct access to all available memory, not just an archaic 640K.

OS/286 and OS/386 propel your programs beyond the limitations of DOS, without forcing you to start all over.

Moving to protected mode is simple because OS/286 and OS/386 give you the same interface as DOS. The hardware is still under your direct control, many 16-bit compilers already generate code suitable for OS/286 and OS/386, and existing highly tuned, machine-specific subroutines running in real mode can be efficiently called from within protected mode. Since most of your own code won't need to be rewritten, your programming investment is preserved. And because OS/286 and OS/386 work with DOS 3.x. they don't affect other programs, device drivers, or TSRs.

In addition to the larger address space offered by protected mode, OS/386 adds 32-bit performance to systems like the Compaq™ 386 which, until now, have been shackled to 8086 emulation.

Dhrystone Benchmark Mic	rosoft C	High C	
	16-bit	32-bit	Scale
IBM AT 6Mhz	793	na	1.0
Compaq 386-16Mhz	2,380	5,837	7.3
HummingBoard-16Mhz	2,777	6,718	8.5
HummingBoard-20Mhz	3,571	8,470	10.7
Vax 8600 (Unix 4.3 BSD,cc)		6,423	8.1
Sun 3/160 (Sun 4.2 3.0A,cc)		3.246	4.1

OS/386 can be customized to give unmodified DOS programs up to 900k on 386 systems, regardless of how many TSR's, networks, disk caches, etc., are installed.



OS/286™ and OS/386™ Features:

- · Huge address space (4GB on the 80386)
- 32-bit performance (80386)
- · No rewriting of device drivers
- Compact code (under 64k)
- · Support for all existing DOS calls
- New INT-21 calls for manipulating segments, invoking real-mode routines and interrupt handlers, and addressing physical memory directly
- Full interrupt vector support
- Powerful debugging: concurrent DOS environment while debugging protected mode programs
- · The ability to run non-Windows programs in a window

A.I. Architects gives you a complete development toolkit:

OS/286 or OS/386 kernel and linker, Symbolic debugger and command processor

Options include:

Managaria (Managaria)

16-bit and 32-bit compilers

High C, Professional Pascal, or F77L FORTRAN

32-bit Assembler

386 HummingBoard™

The basic Developer's Kit is \$495. 32-bit Compilers are \$895. Run time licenses for OS/286 and OS/386 are

available at nominal cost.



Architects, Inc.

One Kendall Square Cambridge, Massachusetts 02139 (617) 577-8052

A.I. Architect's HummingBoard™ is a high performance 386 coprocessor for the PC-XT, AT and compatibles available with the 80387 and 2-24 Mbytes of RAM.

OS/286, OS/386 and HummingBoard are trademarks of A.I. Architects, Inc., Compaq Deskpro 386 is a trademark of Compaq Computer Corp., High C and Professional Pascal are trademarks of Metaware, Inc., F77L FORTRAN is a trademark of Lahey Computer Systems, Inc., Microsoft and MS-DOS are trademarks of Microsoft Corp., VAX 8600 is a trademark of Digital Equipment Corp., Sun 3/160 is a trademark of Sun Microsystems, Inc., Unix is a trademark of AT&T. FILE COMPARISONS (continued from page 30)

No long-enoug

file1

В

A

В

C

When the ends of the file sections are reached, the longest sequence is known and information about the sequence is reported.

The modification to this algorithm allows the searching to stop if a sequence of N matching lines is found, realizing that it might not be the longest sequence that would be discovered if the searching were allowed to continue to the ends of the sections.

		o ortas or the poetions.
jh value:	Long-enough value=2:	
file2	file1	file2
A 3rd sequence	A .	A 2nd sequence
A 2nd sequence B	A B	A 1st sequence
A B 1st sequence C	- A B C	A B 3rd sequence C
= A 4th sequence	- A	A 4th sequence

Figure 6: With the "recursive longest matching sequence" algorithm, the use of a long-enough value often finds exactly the same sequences of matching lines although the discoveries may occur in a different order.

This allows the searching to end prematurely (before the longest sequence has been assured) and can save considerable time. N is called the "long-enough" value. The effects of the long-enough value can be examined by choosing some test pairs of files and comparing the behavior of the algorithm when a longenough value is used and when one is not used. Quite often, the use of a reasonable long-enough value will find exactly the same sequences of matching lines (although the discoveries may occur in a different order), thus producing an identical report of the differences but with a significant improvement in speed (see Figure 6, page 32). In fact, the use of a reasonable long-enough value allows this algorithm to perform in essentially linear time for typical cases, overcoming the previous worry of time efficiency.

The long-enough value is a parameter that you can specify. To determine a good value for your purposes, first guess at the length of the longest

Delta Files and User Reports

A file comparison utility is a versatile tool for a range of situations. It is useful to partition these situations into two distinct cases.

In the most common case, a file comparison is performed so that the differences between two versions of a text file can be inspected visually. The differences are usually expressed as the changes, insertions, and deletions that can be applied to one file to make it identical to the other file. In this case, the primary job of the comparison is to produce a concise and readable report of the differences.

In the course of editing, a file comparison can be used in this way to highlight the differences between a previous version of a file and the current version. Valid modifications can be verified, and spurious edits can be detected. As another example, if a new version of a program is produced, a partial test of its integrity could include a file comparison of its output with the output from a previous version of the program that is known to be correct. If the two out-

puts compare favorably, the new program passes this integrity test. If they do not compare favorably, another file comparison can be used in the debugging process to highlight the changes between a version of a source code file that is known to work and the version that does not work.

In the second case, a file comparison is performed to generate a delta file-a file that contains a report of the differences between the two files. If the file comparison is thought of as comparing an old file with a new file, a backward delta file is designed so that it contains all the information necessary to recreate the old file, given the new file. A forward delta file is designed to be able to recreate the new file, given the old file. In either case, one of the original files can be eliminated without loss of information. If the delta file is smaller than the file it allows to be eliminated, this will result in a savings of disk space. The primary job of a file comparison in this case is to produce a compact delta file.

This use of a file comparison utility is particularly common in version control systems that maintain multiple historical versions of source code files. Only the current version of a source code file is saved, whereas a backward delta file is saved for each historical version. Any historical version can be recreated by applying the appropriate delta files to the current version of the file. The savings in disk space can be tremendous. (Alternatively, some version control systems save the first version of the file and the subsequent forward delta files.)

This usage is also common in telecommunications applications where a file at one or more remote sites has to be updated from a host. A forward delta file is created on the host by comparing the new file with the old file (a copy of the file that exists at the remote site). If the delta file is small, it is often more efficient to transmit the forward delta file and apply it to the old file than it is to transmit the new file in its entirety.

has three features professional programmers can't live without.

		Personal Computers Running the DOS: Operating System
Microsoft. C		
Optimizing Compiler		
Search Ru	n Watch Options Language Calls	Help F8=Trace F5=Go
	*a Add Watch Ctrl+W	AX = 8882
ter CTL *c	tl Watchpoint	BX = 11F8 CX = 0081
short f		of files on DX = 0036
-11.11	Delete All Watch	SP = 11F4
ctl->de	burt o	out with no BP = 11FA SI = 12E8
	go find flags and files on cmd line	
while (argc)	DS = 5579
{	/* first uppercase the arg to sim	ES = 5579 lify parsin SS = 5579
	upper (*++argu);	CS = 5370
	/* check command line for flags */ if (**argv == '-')	
	<pre>{ /* interpret flags if four</pre>	nd */ EI PL
	switch (*++*argu)	NZ NA
		PE NC
		T SS:11FF
	M	icrosoft

Speed.

Fast Execution Speed.

	Microsoft® C 4.0	Microsoft C 5.0
Sieve (25 iterations)	5.7	3.3
Loop	11.0	0.0*
Float	19.9	0.1
Dhrystone	22.8	19.1
Pointer	14.2	7.4
 New optimizations get 	enerate the fastest co	de:

- Inline code generation. NEW!
 Loop optimizations: NEW!
 - - –Loop invariant expression removal. NEW!
 - -Automatic register allocation of variables. NEW!
- -Elimination of common sub expressions.
- Improved constant folding and value propagation.
- Improve constant biding and value propagation.
 Fine tune your programs for even greater speed:
 —Coding techniques for writing the fastest possible programs are included in the documentation. NEW!
 - —Segment Allocation Control:
 - -Group functions into the same segment to get faster NEAR calls. NEW!
 - Specify which segments receive variables to yield faster NEAR references. NEW!
- Uses register variable declarations -Mix memory models using NEAR, FAR & HUGE

Benchmarks run on an IBM® Personal System/2.™ *Time is negligible

Speed.

Fast Compilation. Fast Prototyping.

Microsoft C Version 5.0 includes QuickC,™ which lets you edit, compile, debug, and execute in an integrated environment. It's ideal for prototyping.

- In-memory compilation at over 10,000 lines/ minute. NEW!
- · Built-in editor with parentheses, bracket and brace matching.
- •Use the integrated debugger to animate through your program, add watch variables and set dynamic breakpoints. NEW!
- MAKE file is automatically generated for you. Simply indicate the modules you want to use, then MAKE recompiles and links only those modules that have changed. NEW!
- Full C 5.0 compatibility:
 - —Completely source and object code compatible.

 - -Emits CodeView®-supported executables.
 -Identical compile/link command line switches.

And speed.

Fast Debugging.

Microsoft C Version 5.0 includes Microsoft CodeView, our source-level windowing debugger that lets you debug more quickly and thoroughly than ever before.

- · Debug larger programs:
- Debug through overlays created by the Microsoft overlay linker. NEW!
- -Expanded Memory Specification (EMS) support. NEW!
- Fast debugging through precise control of your program execution:
- Access source level and symbolic debug information from your Microsoft C, FORTRAN, and Macro Assembler programs. NEW!
- View your source code and assembly simultaneously.
- Watch the value of variables change as you execute.
- -Set conditional breakpoints.
- Animate or single step through your program.
 CodeView brings you as close as you've ever been to your hardware:
- —Swap between your code and output screens.
- Watch your registers and flags change as your program executes.

Microsoft

C 5.0 will be available soon. If you purchase Microsoft C 4.0 after June 1, 1987, we'll give you a C 5.0 upgrade. Free. For your free information packet, call: (800) 426-9400.

FILE COMPARISONS

(continued from page 32)

sequence of lines you can imagine appearing more than once in a typical file. The long-enough value should be at least one larger than your guess. This will help the algorithm to avoid matching the wrong instance when a sequence of lines appears multiple times in a file. If a particular choice of long-enough value produces unsatisfactory difference reports, the algorithm can always be applied again with a larger value. When comparing C source code, I typically choose a generous value of 25, and I rarely have to rerun the comparison.

The "recursive longest matching sequence" algorithm is particularly well suited to take advantage of some common hash code technology as a means of improving time performance even more. In applications that involve repetitive string comparisons, it is often useful to calculate hash codes initially for all the strings. Then, the hash codes are compared instead of the strings themselves. The comparison of two hash code values is much quicker than is the comparison of two strings. If the hash codes are not equal, the strings cannot possibly be the same and need not be compared. If the hash codes are equal, only then must the strings be compared to prove or disprove their equality.

The performance benefits are even more dramatic when hash codes are used with the "recursive longest matching sequence" algorithm. When searching for the longest sequence of matching lines, strings do not have to be compared every time a pair of matching hash codes is found. Instead, strings only have to be compared once a sequence of matching hash codes is found that is longer than the longest sequence yet found.

The time efficiency can be improved even further if a hash code table is maintained for each file. The table should consist of an array that contains as many elements as there are possible hash code values. Each element of the array should consist of a linked list of line numbers for lines whose hash code values are equal to the array index. This table

can easily be created by processing each line in the file, calculating its hash code value, and adding its line number to the proper linked list. Now, while searching for the longest sequence of matching lines by examining pairs of starting line numbers, the number of candidate pairs can be greatly reduced. For any given line in one file, only those lines in the other file that have the same hash code value (as can be easily determined from the file's hash code table) need to be considered.

A basic C implementation of the "recursive longest matching sequence" algorithm is shown in Listing One, page 54. Its simplicity, combined with a long-enough value modification and some clever use of hash codes, makes it a viable solution to the file comparison problem. It is suitable for both delta creation and visual inspection purposes.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext.

216. Please specify issue number and format (MS-DOS, Macintosh, Kaypro).

You can also purchase a full-featured executable version of this algorithm from Stepping Stone Software, P.O. Box 2887, Ann Arbor, MI 48106 for \$30. The available format is MSDOS 51/4-inch DSDD.

Bibliography

Heckel, Paul. "A Technique for Isolating Differences Between Files." *Communications of the ACM*, vol. 21, no. 4 (April 1978): 264–268.

Hirschberg, D. S. "A Linear Space Algorithm for Computing Maximal Common Subsequences." *Communications of the ACM*, vol. 18, no. 6 (June 1975): 341–343.

Wagner, Robert A.; and Fischer, Michael J. "The String-to-String Correction Problem." *Journal of the Association for Computing Machinery*, vol. 21, no. 1 (January 1974): 168–173.

DDJ

(Listing begins on page 54.)

Vote for your favorite feature/article. Circle Reader Service **No. 2**.

FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of version 1.1 of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is Bell Labs' answer to Ada and Modula 2. C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: \$195

Requires IBM PC/XT/AT or compatible with 640K and a hard disk. **Note:** C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for \$195:

- The full AT&T v1.1 C++ translator.
- · Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- · Installation guide and documentation.
- 30 day money back guarantee.

To order:

send check or money order to:

GUIDELINES SOFTWARE P.O. Box 749 Orinda, CA 94563

To order with Visa or MC, phone (415) 254-9393. (CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.

Call or write for a free C++ information package.

TASTITIES Everything! Unlocks Everything o



turn this into this!

MASTER*KEY No Other Product Comes Close!

An EXPERT may not know the solution, but always knows where to find it.

MASTER-KEY HELPS ANYONE solve those confusing and frustrating software puzzles more rapidly and easily than any other software available, at any cost! It gives you know-how within hours that may otherwise take years of experience. Create a new program from an old one. DON'T REINVENT THE WHEEL!

MASTER*KEY - Smart!

MASTER-KEY is an intelligent self-documenting MS-DOS reverse assembler. Its sophisticated produced self-wiftly race through massive and baffling object code files to effortlessly discover potential trouble spots.

MASTER*KEY - Educational!

YOU DON'T NEED TO KNOW ASSEMBLY LANGUAGE! MASTER+KEY will take any program from your IBM-compatible computer and return fully-documented, easily-understood assembly language source code (Microsoft MASM 4.0 compatible).

MASTER*KEY - Easy To Use!

MASTER-KEY works both automatically from the DOS command line or interactively from menus similar to Lotus Corporation's 1-2-3 or Symphony. No need to remember any new commands or continually refer to a manual. Use it immediately!

Minimum System Requirements:

256K + 8088/8086/80186/80286/80386 PC MS-DOS or PC-DOS 2.0 + One 360K DSDD Floppy Drive (IBM PC Format)

MS-DOS is a trademark of Microsoft. PC-DOS is a trademark of IBM.

0	EB 76 EO	18 65 3D	49 72 36	6E 73 01	69 72	6F 05	6E	OD-OA	24	50	B4	30	CD	21	86	k.Incorrect DOS version@P4ON!. '=6.r.=v.:4. N!N Xk/	0
0																	0

H00100: JM	IP Sh	ort HOO	114				;00100	EB18	<u></u>
DE		ncorrec	t DOS	versio	n"		;00102	496E636F7	27265
DE		The state of the s					;00117		
DE							;00118		
DE			284				;00119	24	
H0011A: PU	SH AX						:0011A	50	P
MO	V AH	,30h					;0011B	B430	_0
IN				;1-DO	S_Ver_	Number	;0011D	CD21	_1
		, AL					;0011F		
CN		,0136h						3D3601	=6_
JB		012B		;			;00124		r_
CH		,020Ah						3D0A02	
JB H0012B: MC		0134		;			;00129		v_
HOO12B: NO		,0102h						BA0201	
IN				.1 D.	1	String	;0012E		
IN				TEDM	spray_	lly:20h			_1
,	20			; IERR	_norma	11y.20h	;00132	CDZO	-
H00134: P0	P AX						:00134	58	x
JM	P Sh	ort HOO	166				:00135		_/
;	•						-11.07 (1)		
MASTER*KEY	XREF -	PROGRAM	.XRF						Page 1
0102h			121	2F5	301	320			
020Ah			126						
озсвь		:	12B						
1-Display_		:	130	591	610				
1-DOS_Ver_	Number	:	11D			NO	TC. The		maa ia bu
H00100		:	100			NU		cross-refere	
H0011A		:	100	11A			mem	ory location	n within
H0012B		:	124	12B				rogram file	
H00134		:	129	134			me h	nogram me	4
H00166		:	135						
TERM_norma	11y:20h	:	132			NO	IE: The	output is to	tally
							Micr	osoft MASN	/I-compatib

MASTER*KEY will guide you step by step to:

- 1. Help you learn assembly language, if you desire.
- 2. Discover how any program runs or why it doesn't.
- Alter or remove unwanted object code from any program.
- Incorporate routines from compiled programs into other assembly language, Basic, C, or Pascal programs.
- Make software more compatible with your computer. Be certain a questionable program won't damage your system BEFORE you run it.
- Modify software to operate with other versions of DOS.
- Customize your COMMAND.COM or other executable program directly or by reassembling your altered MASTER*KEY source code.

Order Now! Just \$7995

Phone orders accepted on MC or VISA Send checks to:

Sharpe Systems Corporation 2320 E Street, La Verne, CA 91750

\$82.45 (includes \$2.50 shipping) \$87.65 in California (includes tax & shipping) C.O.D. orders add \$2.00

(714) 596-0070

Dealer/Distributor Inquiries Welcome

"How to protect your software by letting people copy it."

By Dick Erett, President of Software Security



Inventor and entrepreneur, Dick Erett, explains his company's view on the

protection of intellectual property.

crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment Hard Disk Installation: Simply copy program disk to hard disk using DOS Command - Copy A:*.* C:

Program Back-ups: You may make as many copies of the program diskette as you wish.

Data Back-ups: Use normal back-up and restore commands, including backing up sub-directories containing program files.

Tea Networks: This product may be Networks. Follow the same installation and on page 102 of this manual. The Block and on page 102 of this manual. The Block and on page 102 of this manual operation of any

Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.

of the market, or take a stand against the theft of your intellectual property.

"... giving your software away is fine..."

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCKTM.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

In reality, the only people who could object are those who would like the option of stealing your company's product.

"... eliminating the rationale for copy-busting..."

Since The BLOCK allows a user to make unlimited archival copies the rationale for copy-busting programs is eliminated.

The BLOCK is fully protected by federal patent law rather than the less effective copyright statutes. The law clearly prohibits the production of work-alike devices to replace The BLOCK.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

"...possibilities... limited only by your imagination..."

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."



870 High Ridge Road Stamford, Connecticut 203 329 8870

06905

The XOR Chain Revisited

by Bennette R. Harris

n "The XOR Chain" (DDJ, June 1987), David Cortesi showed how to use the exclusive-OR (XOR) operation to compress two pointer items into one link field in a doubly linked list. This trick is just one of many that have been used for this purpose over the years. In this article I'll describe some similar tricks for doing the same thing, discuss their relative advantages and disadvantages, and then present a package of C functions that use the XOR technique to manipulate binary trees of arbitrary depth without using stacks.

The Tricks of the Trade

Cortesi's article discussed a few important properties of the XOR operation:

$A \times OR 0 = A$	(1
$A \times A = 0$	(2
$(A \times C) \times A = C$	(3)
(A XOR C) XOR C = A	(4)

Properties 3 and 4 make it possible to store two pointers, A and C, in a single field as A XOR C and then recover either one of the two provided the other is known. Thus, if the addresses A, B, and C point to three nodes in sequence, and if the link field W of node B is set so that:

Bennette R. Harris, 231 S. Janesville St., Whitewater, WI 53190. Dr. Harris is an assistant professor of mathematics and computer science at the University of Wisconsin-Whitewater. He teaches in the Management Computer Systems program and he also serves as a consultant both privately and through the university's Small Business Development Center.

Managing trees without stacks

W = A XOR C

then W XOR A yields C, permitting travel in one direction through the list, whereas W XOR C yields A, permitting travel in the other direction.

The basic purpose of a storage technique such as this, as the preceding example illustrates, is to permit travel in two directions through a data structure while storing only a single link field. It assumes that you entered the data structure at a predefined entry point (such as a head or root node) and that an adjacent pair of nodes are referenced at any given point in time.

It is not essential that you use the XOR operation for combining the addresses—you can use almost any reversible operation. For example, provided the addresses are in a range that won't generate overflow errors, you could use the operations:

$$W = A + C$$

$$C = W - A$$

$$A = W - C$$

to move about in the data structure. This observation might be handy if you wanted to use this technique in an environment in which XOR was not available. The chief advantages of using XOR are its speed and never

getting results that are out of range.

Any such technique exchanges a fairly minimal amount of computation for a reduction in the amount of storage required to hold pointers. All, however, suffer from the problem that the data stored in a link field is "unnatural"-it does not actually represent a pointer to any one node of the data structure. Instead, its contents are encoded, and the information can only be recovered using an adjacent node's pointer as a key. If a portion of the data structure becomes corrupted, the data structure as a whole becomes practically worthless because it's virtually impossible to make any sense out of links in nodes located beyond the point of corruption. Also, such link fields are hard to trace from a dump of the data structure.

You can achieve the same space savings while maintaining "natural" links by reversing the link field of the current node to point backward to the previous node as you go down the data structure and then reversing the link again as you go back up. This way, the link field remains a true pointer to nodes of the data structure. Supposing that addresses A, B, and C pointed to three nodes in sequence, the link field W of node B would initially be set so that W = C. Then, as node B was accessed traveling forward, a series of statements similar to the following would be executed:

NextNode = WW = PrevNode

On the return back through the structure, of course, the link field of

node B would be reset using the same statements. For an example of this technique in action, see Allen Holub's discussion of nonrecursive tree traversal in C Chest, *DDJ*, July 1986.

This traversal technique exchanges the XOR computations for the overhead required to reverse the link fields. Although links are always true pointers to interesting places, a disadvantage in this case is that the data structure must be exited from completely in the reverse manner to that in which it was entered in order to restore pointers to a useful state. This is undesirable if the data structure is a large linked list, although it's perhaps acceptable in a tree. A second undesirable feature of this method is that, if a program or computer were to go down while in the middle of such an operation, the data would again be corrupted, although not irreparably so. Because the likelihood of such problems seems to be relatively high in a microcomputer environment, this is usually not acceptable. Finally, observe that the data must be accessed twice-once to read it and once to write it again with the new link. In some applications this could represent a significant amount of overhead.

Given the alternatives described here, the XOR chain stands out as one of the safest link compression techniques because the link fields are not changed in traversing the data structure.

Climbing the Tree

To keep matters simple, I will focus on binary trees in this article. Those of you who have experience with more general tree structures will be able to make the necessary adjustments.

In a binary tree, each node contains two link fields—a left link and a right link—which point to the two children of the current (parent) node. Routines for manipulating trees often make use of stacks or recursive programming techniques to maintain the back-pointers required by these routines. In many cases the necessary stack space must be set aside in advance, limiting the potential depth of the tree. With the XOR chain technique, the link fields can be coded so they can be used to travel either from parent to child or from

child to parent, much as in the case of linked lists. This makes it possible to implement a binary tree without setting aside a stack to maintain backpointers for traversing the tree.

The only reason why implementing a tree is not trivial has to do with the fact that a node has two children. Suppose you wish to visit the nodes in a tree in the fashion known as "inorder," visiting the left child, then the parent, and finally the right child. When you return from a child to its parent, you need to know whether the child was a left child or a right child. If the child was a left child, then the parent must be visited

The chief advantages of using XOR are its speed and never getting results that are out of range.

next. If the child was a right child, then the parent has already been visited, and it is time to return to the parent's parent. This is not hard to determine if the links of the parent can be matched with the node just visited, but when these links have been coded as described earlier, it is not so simple.

There are at least two ways out of the difficulty. One is to include an extra bit in each node, indicating whether it is a left or right child. This extra bit is a small sacrifice for the convenience of the doubly linked pointers but still might be wasteful in a large data structure. The other way is available only if the binary tree is sorted. If the tree is sorted so that (for example) the key field of the left child is less than the key of the parent, which is in turn less than or equal to the key of the right child, then whether a node is a left child or a right child can be determined by comparing the child's key field with that of its parent. Because binary trees are frequently used to index other data structures, this criterion is often met.

Defining an Item

The structure of an item (or node) in the tree is defined as follows:

```
struct Item
{
  int key;
  unsigned llink,
    rlink;
  /* other stuff */
};
```

The key field contains the key based on which the tree is sorted. Here, I have chosen *key* to be of type *int*, but you could use other types, depending on your application. The link fields contain the XORed linking addresses. The "other stuff" is application specific.

I assume that *Item*s are created and destroyed as in Cortesi's article:

```
extern struct
Item *MakeItem();
extern void
DropItem(i) struct Item *i;
```

Defining a Tree

A binary tree consists of a collection of zero or more *Items* that satisfy a hierarchical relationship. If a tree is not empty, then it has a unique root item that may have zero, one, or two children. Each child item is the root of a subtree that is itself a tree. The root item of a tree is an *Item* just like any other in the tree, except that it is not the child of any other item. It is convenient not to store any data in the root item so that an empty tree consists of a root item with no children. This makes it possible to define a function to test for an empty tree:

I'm assuming that the tree is sorted in-order fashion—that is, the key of the left child must be less than the key of the parent and the key of the right child must be greater than or equal to the key of the parent. I'm giving the root item an artificially high key so that the data-containing items are all found in the root's left subtree. The root should be initial-

XOR CHAIN (continued from page 37)

ized with:

```
root->key = MaxKey
```

where *MaxKey* is an appropriately defined value greater than any other key in the tree.

Traversing the Tree

To travel within a tree, you must enter at its root and then move up or down the tree from child to parent or parent to child. At any point within the tree, your position is described by a child-parent pair of pointers that contains the addresses of the item (child) being visited and its parent. As in Cortesi's article, these addresses are stored, together with the address of the tree's root, in a record called a *Scan*:

```
struct Scan
{
    struct Item *parent,
        *child,
        *root;
};
```

Before it can be used, a *Scan* must first be associated with a particular tree by having its root field set:

```
void Associate(s,r)
struct Scan *s;
struct Item *r;
{
```

```
s->root=r;
```

Once a *Scan* is associated with a particular tree, it can be positioned at the root of the tree to begin the tree traversal:

```
void ToRoot(s)
struct Scan *s;
{
   s->parent = NULL;
   s->child = s->root;
}
```

Because the root item of a tree has no parent, you can very easily test to see if a *Scan* is at the root of the tree:

```
int AtRoot(s)
struct Scan *s;
{
    return(s->parent == NULL);
}
```

This function can be used to detect the end of a sequential walk through the tree because the root's key value is greater than all other keys in the tree and so it will be visited last.

Moving the Scan

Three fundamental actions must be described for moving around within the tree: a move from the parent to the left child, a move from the parent to the right child, and a move from a child to its parent. The first two are easy and are described by the code in Example 1, below.

The third move is more interesting (see Example 2, below). You must be able to recover the parent's parent to implement a move from the child to its parent. To do so, you must determine whether the child is a left or right child so you can XOR with the proper link field. Naturally, the *Is-Left* function needs to be modified if the necessary position information is coded differently in the tree, as would be the case for height-balanced trees that allowed nonunique keys.

Notice that the procedures for these three actions are not recursive and do not require the use of a stack for back-pointers.

Once the three fundamental actions have been described, you can write procedures that will process the tree sequentially in either direction as if it were a linked list. I'll do an in-order traversal here—that is, visit the left subtree recursively, then the root, then the right subtree. First, you need to be able to move forward and backward within the tree's list, as shown in Example 3, page 39. Both these procedures are designed to stop at the root item to make it easy to test for the ends of the tree's list.

To start sequential processing, you first have to locate the head or tail of the list:

```
void ToHead(s)
struct Scan *s;
{
    ToRoot(s);
```

```
void GoLeft(s)
struct Scan *s;
{
    struct Item *i;

    i = s->parent ↑ s->child->llink;
    s->parent = s->child;
    s->child = i;
}

void GoRight(s)
struct Scan *s;
{
    struct Item *i;

    i = s->parent ↑ s->child->rlink;
    s->parent = s->child;
    s->child = i;
}
```

Example 1: Moving from the parent to its children

```
int IsLeft(s)
struct Scan *s;
{
  return(s->child->key < s->parent->key);
}

void GoParent(s)
struct Scan *s;
{
  struct Item *i;
  if (IsLeft(s))
   i = s->parent->llink \( \) s->child;
  else
   i = s->parent->rlink \( \) s->child;

s->child = s->parent;
s->parent = i;
}
```

Example 2: Moving from a child to its parent

```
while (s->child->llink !=
                          s->parent)
    GoLeft(s);
void ToTail(s)
struct Scan *s;
  ToRoot(s):
  GoBak(s);
```

Inserting an Item

Unlike the case for a linked list, new items cannot be inserted at arbitrary places within the tree; instead, new items must be added to the tree in such a way that the tree remains sorted.

In the insertion procedure shown in Example 4, below, the scan is passed to identify the tree in which the item is to be inserted. The tree is traversed (not sequentially!) until the proper insertion point for the item is

```
found. The item is then inserted as a
leaf of the tree, with no children of
its own. First, the appropriate link
field of the item's parent is set to
point to the new item (using XOR),
and then the item's links are set to
point back to its parent. Finally, the
scan is returned with the new item
as the current active child.
```

Deleting an Item

Deleting an item from a binary tree is much more complicated because the resulting items must still maintain the binary structure and sorted arrangement of the tree before the item was deleted.

In my implementation, if the item has only one child, then I delete it by pointing its parent to the nonempty subtree, and vice versa. If the item has two children, I delete the item by removing it and replacing it with the greatest item in the left subtree. This item will have at most one child (a

left child), and so the item can be repositioned with a minimum of manipulation of the link fields. Thus, procedure the used here must handle four cases:

1. The item to be deleted has no children (an item

its link fields are equal to each other because both would point back to the item's parent).

2. The item has a right child but no left child (an item has no left child if the left link equals the pointer to the item's parent).

3. The item has a left child but no right child.

4. The item has both a left and a right child. There are two possibilities:

a. If the left child has no right child of its own, then the left child replaces the item and picks up the item's right subtree.

b. If the left child has a right child, you travel down through right children as far as you can go to locate the greatest item less than the item to be deleted.

These cases are more difficult than usual, even for a tree delete routine, because not only must parent pointers be adjusted to point to their new children but also child pointers must be adjusted to point back to their new parents. This means testing to make sure the children exist before accessing their link fields.

When the deletion procedure (see Listing One, page 66) is called, the node to be deleted is pointed to by s->child. The scan is set back to the parent (by a call to GoParent) before the routine returns so that no scan is ever returned with a NULL child (because the child item of the scan is the has no children if | item being visited). In fact, GoParent

```
void GoFwd(s)
struct Scan *s;
 if (s->child->rlink!=s->parent)
   GoRight(s);
   while (s->child->llink != s->parent)
    GoLeft(s):
 else
   while (!AtRoot(s) &&!IsLeft(s))
    GoParent(s);
   if (!AtRoot(s))
    GoParent(s);
void GoBak(s)
struct Scan *s;
 if (s->child->llink!=s->parent)
   GoLeft(s);
   while (s->child->rlink!=s->parent)
     GoRight(s);
 else
   while (!AtRoot(s) && IsLeft(s))
    GoParent(s);
   if (!AtRoot(s))
     GoParent(s);
```

Example 3: Moving backward and forward within the tree's list

```
void Insert(i,s)
struct Item *i;
struct Scan *s;
 ToRoot(s);
 while (s->child != NULL)
   if (i->key < s->child->key)
    GoLeft(s);
   else
    GoRight(s);
 if (i->key < s->parent->key)
   s->parent->1link \uparrow = i;
 else
   s->parent->rlink 1= i;
  i->1link = s->parent;
  i->rlink = s->parent;
  s->child=i;
```

Example 4: The insertion procedure

XOR CHAIN

(continued from page 39)

is called early in the procedure, before any pointers are rearranged.

Conclusion

As with any arbitrary binary tree system, it is possible for the tree to become quite unbalanced after several insertions or deletions, degrading the performance of the system. A good question is whether the same idea could be woven into an AVL height-balanced tree system such as that discussed by Allen Holub (DDJ, August 1986, page 20). The answer is no, not in the form presented here. My routines rely heavily on the fact that the key of any item is strictly greater than the key of its left child. This cannot be guaranteed in an AVL tree if nonunique keys are allowed. In that case, it would be necessary to use the alternative approach I mentioned earlier: code a bit in each item, marking it as either a left child or a right child. With this modification, height balancing is possible.

Another interesting implementation would involve B-trees and other generalized tree structures. Such implementations are possible with unique keys, or with a few code bits added to each tree item, and can be coded in much the same way as the examples shown here.

Finally, I should say a few words about another nonrecursive traversal method—namely, threaded trees. The main problem with this technique is maintaining the threads as items are added to and deleted from the tree. By their very nature, threaded links are links to items that usually are not adjacent to the item being added or deleted, and so there are very few easy cases in the add and delete algorithms. I will leave it to you to determine which has the more substantial overhead.

A wise instructor once said, "Ninety-nine percent of all programs using recursion could be written more effectively without it." The XOR chain technique makes it possible to apply this maxim to tree structures as well. With a little thought and creativity, you should be able to customize these ideas to fit your needs.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600, ext. 215. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

Bibliography

Holub, Allen. "C Chest: Nonrecursive Tree Traversal." DDJ 117 (July 1986): 18-20. There's a bug fix in DDJ 123 (January 1987): 103.

Holub, Allan. "C Chest." DDJ 118 (August 1986): 20-29.

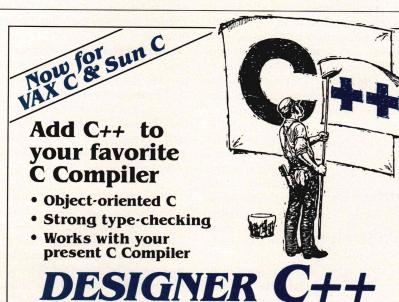
Knuth, Donald E. The Art of Computer Programming, Volume 3: Sorting and Searching. Reading, Mass.: Addison-Wesley, 1973.

Singh, Bhagat; and Naps, Thomas L. Introduction to Data Structures. St. Paul, Minn.: West Publishing Co., 1985.

DDJ

(Listing begins on page 62.)

Vote for your favorite feature/article. Circle Reader Service No. 3.



BENEFITS:

- You can incrementally add C++ features to C (switch-selectable)
- Makes C more suitable for very large programs
 - more sophisticated applications
- ► Works with Sun's dbxtool
- ► Works with the C Compiler you now use
- ➤ More reusable code
- ▶ Resilient and bug-free code

The only commercially available C++ customized to operate on PC's, micros, minis, and mainframes with popular C compilers, including:

VAX C ULTRIX C SUN C MICROSOFT' LATTICE

GREEN HILLS APOLLO XENIX

HP-9000 UNISOFT

*Lattice and Microsoft versions of Designer C++ are known as Advantage C++

We Specialize in: Cross/Native Compilers: C. Pascal, FORTRAN, Ada. LISP.—
Assemblers/Linkers.— Symbolic Debuggers.— Simulators.— Interpreters.— Profilers.— OA Tools.— Design Tools.— Comm.
Tools.— OS Kernels.— Editors.— VAX & PC
Attached Processors and more
We Support. 680xx, 80x86, 320xx, 68xx,
80xx; Clipper, and dozens more

FFATURES.

checking

operators

functions)

 Fully compatible with AT&T C++ standard

Optional strong type

 Overloading of function names and

Dynamic typing (virtual

User-defined implicit

type conversion

Data abstraction



60 Aberdeen Ave., Cambridge, MA 02138 (617) 491-4180 1219 Morningside Drive, Manhattan Beach, CA 90266 (213) 546-5814 (CA only)

Designer C++ is a joint trademark of XEL, Inc. and Glockenspiel. Ltd of Dublin. Ada is a trademark of the U.S. Government (AJPO), Advan-lage C++ is a trademark of Lifeboat Associates. Inc. Other trademarks are acknowledged to DEC, Lattice, Microsoft & Sun Microsystems, Inc.

THEADA WORLD HAS CHANGED.

VALIDATED

Meridian's AdaVantage™ v2.0 compiler is a complete implementation of the Ada® language that has been validated on the IBM PC/XT, IBM PC/AT, and the Zenith Z-248. Most of the representation clauses and implementation-dependent features are also available including

- ▲ pragma pack
- ▲ size specification
- ▲ task storage size
- ▲ fixed point small
- ▲ record representation clauses
- ▲ address clauses
- ▲ package system
- ▲ representation attributes
- ▲ pragma interface
- ▲ unchecked storage deallocation
- ▲unchecked type conversions

The compiler includes a complete set of utilities for managing the Ada program library and all of the standard packages including text_io, system, and calendar.

BEST PRICE PERFORMANCE

The Meridian AdaVantage compiler demonstrates the best price/performance of any PC Ada compiler. The compiler sells for \$795 in single quantities and it compiles about 1000 lines per minute on an IBM PC/AT.

In addition to the production compiler, two other versions are available for general training and student use. The AdaTraining™ compiler sells for \$395 and is intended for corporate and university educational environments. The AdaStarter™ compiler, priced at \$129, incorporates all of the features of the AdaVantage production compiler, with certain limitations on the number of library units and the number of lines per compilation unit allowed. The full price of the AdaStarter compiler can be applied to a later purchase of the AdaVantage production compiler.

ADDITIONAL PRODUCTS

Two optional packages, priced at \$50 each, that provide DOS environment support and miscellaneous utility routines are currently available. A source-level debugger and Ada editor will be available this Fall.

CONFIGURATION

The compilers all run in a standard PC configuration with 640K of memory, a hard disk, and DOS v2.1 or higher.

To order today, or get more information, call toll-free 1-800-221-2522.



23141 Verdugo Drive, Suite 105 Laguna Hills, CA 92653 800/221-2522 (outside Calif.) 714/380-9800 (inside Calif.) Telex: 650-268-0547 MCI

Ada is a registered trademark of the U.S. Government (AJPO). AdaVantage, AdaTraining, and AdaStarter are trademarks of Meridian Software Systems, Inc. References to other computer systems use trademarks owned by the respective manufacturers.



Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today, You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. \$185.

Special: VEDIT (single file, no windows) for CP/M-\$49.



Call 1-800-45-VEDIT for FREE Fully Functional Demo Disk

- Fully Network Compatible
- Call for XENIX-286 version
- 30 Day Money-back guarantee

Compare Features and Speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	Unlimited
Multiple file editing	20+	2	No	20+
Windows	20+	2	No	20+
Macro execution window	No	No	No	Yes
Pop-up menus	No	No	No	Yes
Execute DOS commands Automatic processing of	Yes	Yes	Yes	Yes
Compiler errors	Yes	No	No	Yes
"Cut and paste" buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
Convert to/from WordStar	No	No	No	Yes
On-line calculator	No	No	No	Yes
Configurable Keyboard	Hard	No	Hard	Easy
43 line EGA support	Yes	No	No	Yes
Manual size/index	250/No	42/no	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec





VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of the The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others. *Demo disk is fully functional, but does not readily write large files.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821

WINDOWS FOR DATA™

The first choice of professional C programmers

"Windows for Data is the best programming tool I've ever used. It's the most flexible I've seen. Whenever I've wanted to do something, I've been able to find a way."

> Steven Weiss, Stratford Systems

Professionals choose our tools because they are designed, crafted, and supported for professionals. Here at Vermont Creative Software, we understand that performance and pleasure in programming derive from more than a long list of functions. Windows for Data provides:

PROFESSIONAL FLEXIBILITY:

Our customers repeatedly tell us how they've used WFD in ways we never imagined - but which we anticipated by designing WFD for unprecedented adaptability. Virtually every capability and feature can be modified to meet special needs. You will be amazed at what you can do with WFD.

PROFESSIONAL PERFORMANCE:

Screen output is crisp and fast. Windows, menus, and data-entry forms snap up and down from the screen. WFD is built upon and includes Windows for C, the windowing system rated #1 in speed and overall quality in PC Tech Journal (William Hunt, July 1985).

PROFESSIONAL RELIABILITY:

An unreliable tool is worse than no tool at all. VCS products are known in the industry for their exceptional reliability. Ask anyone who owns one.

PROFESSIONAL DOCUMENTA-

TION: Over 600 pages of documentation provide step-by-step explanations for each major application, a reference page for each function, listings of functions alphabetically and by usage, and a fully cross-referenced

index. Extensive tutorials and demonstration programs assist learning.

PROFESSIONAL TECHNICAL SUPPORT: The same expert programmers that develop our products provide prompt, knowledgeable technical support.

PROFESSIONAL PORTABILITY: High-performance versions of VCS products are available for XENIX, UNIX, and VMS, as well as DOS. No royalties on end-user applications.

OUR CHALLENGE AND GUARANTEE

If you have an application where no other tool can do the job, try **Windows for Data**. If it doesn't help you solve your problem, RETURN FOR A FULL REFUND. YOU MUST BE SATISFIED.

Ask for FREE DEMO DISKETTE



Vermont Creative Software

21 Elm Ave. Richford, VT 05476 Telex: 510-601-4160 VCSOFT

Tel.: 802-848-7731

Prices: PCDOS* \$395; XENIX, VMS, UNIX
*PCDOS specify C compiler.

WINDOWS FOR DATA

for DOS, UNIX, VMS ...

The complete windowing data entry, menu, and help system that does the hard job others can't — we guarantee it!

Pop-up data entry windows; field types for all C data types, plus decimals, dates, and times; auto conversion to and from strings for all field types; system and user supplied validation functions; range checking; required, must-fill, and protected fields; free-form movement; multiple-choice field entry; scrollable sub-forms. Branch and nest windows, forms, and menus.

Complete context-sentitive help system with pop-up windows and scrollable text.

Pop-up, pull-down, scrollable, and Lotusstyle menus.

NEW FOR DEBUGGING: Exclusive VCS Error Traceback System automatically identifies the location and cause of program errors. Eliminates the need to code error checks on all function calls! VCS Memory Integrity Checking helps catch those hard-to-detect, memory-corruption errors.

NEW FOR ERROR HANDLING: Install your own error handler to be called whenever a function detects an error.

NEW FORM LAYOUT UTILITY simplifies form design.

Writing MS-DOS Device Drivers in C

by Andy Klein

he MS-DOS/PC-DOS installable device driver facility has been available since Version 2.0 and allows you to add extra devices to your system without making any modifications to DOS itself. You specify devices to be added in an ASCII file called CONFIG.SYS using the command device = xxxxxxxxx.yyy. Installable device drivers allow hardware independence as hardware-specific code is isolated in the driver.

This article shows how to write the functions that implement a device in C. To do this, I discuss the format of a device driver and how to create a driver in this format using (mostly) object code produced by a C compiler. Some of the material is compiler specific and pertains to Aztec C, but it should be portable to other C compilers (even to other high-level languages). The example device driver I present—prndrv.asm in Listing One, page 68-is a simple one that implements a parallel printer and replaces the standard PRN device. I've chosen a simple device deliberately so that I can concentrate on how to construct drivers in C instead of on the details of a specific device and at the same time provide a real, working device driver that you can modify and experiment with on your PC.

Request Header

DOS communicates with its device drivers through a packet called a request header, which is a formatted

Andy Klein, 3801 N. 16th St. #222, Phoenix, AZ 85106. Andy's company, Micro Quantitative Sciences, develops software interfaces for the IBM PC and peripheral devices.

An alternative to assembly language

block of system memory that contains the command code that the driver is to perform, a status word that the driver uses to report the success or failure of the operation back to DOS, and the length of the packet. The header is followed by data needed for the operation, which varies depending on the operation to be performed. For my driver, this variable area can be considered fixed, containing the segment, offset, and number of bytes to transfer. Req_hdr is a C structure that is typedefed in rh.h (Listing Two, page 71) and is used by the driver implementation functions to access the request header data. An important task you need to accomplish is to make the request header addressable by C compiled functions (discussed later).

When DOS calls a driver to perform a task, it does so in two separate stages. The first stage is referred to as the device strategy and occurs when the device is passed a long (segent:offset) pointer to the request header in the es:bx register pair. The device does not perform the request at this stage—it just saves the pointer to the request header. The next stage is called the device interrupt. It receives no parameters; instead, the interrupt function retrieves the previously saved pointer to the request header and then performs the operation indicated by the command code

in the header. It is the interrupt function that actually does the work of the device, and it also has the task of setting things up so that you can call C compiled functions. DOS always calls the strategy function, then immediately calls the interrupt function. Following this pattern, device drivers have two entry points, called the strategy and interrupt routines. These entry points are implemented in prndrv.asm as labels $dev_strategy$ and $dev_interrupt$.

Format of a Driver

DOS device drivers must be in a specific format in order to be incorporated into MS-DOS. This format differs from that of a .COM or .EXE file. Drivers must start with a device header that starts at offset 0h from the start of the file, and all the logical segments of the driver must be in one physical segment, such as a .COM file, because the driver is simply loaded into memory by DOS-it will not get any of the segment fix-ups that an .EXE file receives when loaded. The largest hurdle you'll face when writing your first device driver is to get it to load at boot time without hanging up the computer. Once you get past this point, the rest is cake.

Device Header

The device header's purpose is to provide DOS with the attributes of the device, the offsets of the strategy and interrupt entry points into the driver, and the name of the device or the number of units it controls. The fields of the header are:

- pointer to next header (4 bytes)
- attribute (2 bytes)
- pointer to strategy routine (2 bytes)
- pointer to interrupt routine (2 bytes)

• name or number of units (8 bytes)

The pointer to next header is a 4-byte field that DOS uses to store a pointer to the next driver in the chain of all drivers. These four bytes should be filled with 0xff in prndrv.asm to indicate to DOS that there is only one driver in this file.

The attribute field is bit-mapped as shown in Table 1, page below. My example driver is a character device and has bit 15 set; the others are all 0s.

The name or number of units depends on the driver type. If the driver is a character device, then this field is an eight-character name, leftjustified and padded with spaces. The name assigned here will be the name DOS uses for the device. When character devices are loaded at boot time, installable devices are linked into the chain of all drivers before the default or built-in character drivers. When DOS searches this linked list for a named driver, it stops on the first match, so if you use the name of a default driver as the name of your driver, you replace the default driver with your own. This is exactly what the example driver does, replacing the default PRN device with my PRN device.

Note that there is one exception to this rule—the NUL device cannot be replaced because it is the starting point of the linked list of devices. Block devices do not have names; instead, they are assigned drive letters by DOS in the order in which they are loaded. Unlike character device drivers, the default block drivers are loaded first and cannot be replaced. A block driver can control multiple subunits, and the number of units controlled is entered in this field for block devices with the last 7 bytes of the field filled with spaces. The number of subunits can be overridden by the initialization function for block devices. An example of the use of subunits is a fixed disk controller that can have two fixed disk drives attached.

Device Driver Start-Up Code

In order to write the functions that implement a device in C, a few special things have to be done. First, before any C compiled functions are called, the environment (segment

registers and stack pointer) must be set up to correspond with what the compiler-generated code assumes. Normally this setup is done by the start-up code that is provided with the compiler, which for Aztec C using the small model is in the Aztec function sbegin.asm. All Aztec C compiled functions make an external reference to a function called \$begin, causing the linker to drag it in from the standard library c.lib. \$begin in turn makes an external reference to Croot, which in turn references main. In the normal case, \$begin sets up the environment and

Installable character devices are linked into the chain before the default drivers.

calls *Croot*, which then calls *main*. Other compilers follow the same pattern.

Much of what is done in the compiler start-up functions does not have to be replicated in the driver start-up code, including parsing $argv(\cdot)$, allocating a heap, getting access to the DOS environment variables, and other initializations that are required for some library functions. There is a penalty exacted for this approach, and it is that your C functions cannot

call library functions that require some setup that you have not done—notably, malloc(), free(), and all those functions pertaining to I/O are off limits. Also, variables declared outside functions must be initialized to some value. An essential part of the compiler's start-up that must be incorporated into the device driver start-up code is establishing a stack frame.

In the example driver, the start-up tasks are accomplished by replacing the normal compiler start-up code with the device interrupt function. This function takes care of setting up the segment register to what Aztec C expects. There is no main() because the entry point is the label dev_interrupt in prndrv.asm. It is necessary to provide a function called \$begin to prevent the linker from reporting an error, so prndrv.asm has a function of this name but it is never called. Code generated by earlier versions of Aztec C also has a reference to a function called \$cswt, which is also in prndrv.asm. Users of other compilers will have to replace these functions with whatever their compiler demands. If the source for the startup function for your compiler is available, it is fairly straightforward. (Compiler publishers that don't make this source available significantly limit the applicability of their products for serious system development.) You can discover much of what your compiler does by compiling a few functions and getting the compiler to output the assembly-language result of the compilation.

Device Strategy Function

The strategy function is called with a long pointer to a request header in the *es:bx* register pair. The strategy function does not perform the re-

bit	15	1 = character device,	0=block device
bit	14	1=supports <i>ioctl</i> ,	0=no ioctl
bit	13	1 = non-iвм format,	0=івм format (block device)
		1 = supports output until busy,	0=doesn't (char dev)
bit	12	0 (reserved by Microsoft)	
bit	11	1 = supports removable media	(block device only)
bits	10	-5 reserved, should be 0	
bit	3	1 = clock device,	0=not clock device
bit	2	1=NUL device,	0=not NUL device
bit	1	1=stdout device,	0=not stdout device
bit	0	1 = stdin device,	0=not stdin device

Table 1: Bit mapping of the attribute field

"The Ada programming language shall be the single, common, high order programming language for...

"...all computers that are integral to, physically a part of, dedicated to, or essential in real time to a performance of the mission of weapon systems... used for specialized training, diagnostic testing and maintenance, simulation, or calibration of weapon systems... used for research and development of weapon systems...Use of validated compilers is required...this directive is effective immediately."

—DoD Directive 3405.2, 3/30/87.

"...Defense computer resources used in intelligence systems, for the command and control of military forces...all major software upgrades...all other applications (some exceptions) in keeping with the long range goal of establishing Ada as the primary DoD higher order language...waivers to the policy...shall be strictly controlled and closely reviewed... this directive is effective immediately."—DoD Directive 3405.1, 4/2/87.



New Alsys Toolset For 68000 Ada **Builds Unique Project Environment**

Organizations serious about the 680XO architecture, and serious about working with the government, want a lot more than just validated Ada compilers. They want quality solutions; production quality compilers and quality programming tools.

Just what Alsys offers. Alsys' new 68000 Ada Developer's Toolset includes:

- AdaPROBE, a unique source-level symbolic debugger and program viewer;
- Cross-Referencer, an inter-unit crossreferencing utility;
- Reformatter, a pretty printing tool for reformatting source files to selectable conventions; and
- AdaMAKE, an automatic recompilation facility.

Consider, too, all those special Ada "manager tools" that are part of the Alsus Version 3 compilation system: the Family Manager, the Unit Manager, and the Library Manager.

Together, they implement the new

Alsys Multi-Library Environment that allows teams of programmers to share thousands of logically organized compilation units.

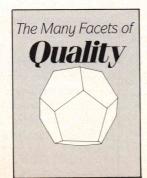
Alsys 68000 compilers are in a class by themselves; highest code quality, maturity, reliability, robustness, superior optimization technology, unexcelled error messages...And now, with the new development tools, they are at the core of an Ada project environment unique in the industry.

Here is our special INTRODUCTORY OFFER. Between now and October 31, 1987, order any of our 68000 Ada compilers and we will include the complete Toolset FREE. AdaPROBE, Cross-Referencer, Reformatter, AdaMAKE.

Ada is Now. Alsys solutions are now.



In the US: Alsys Inc., 1432 Main St., Waltham, MA 02154 Tel: (617) 890-0030 In the UK: Alsys Ltd., Partridge House, Newtown Rd., Henley-on-Thames, Oxon RG9 1EN Tel: 44 (491) 579090 In the rest of the world: Alsys SA, 29 Avenue de Versailles, 78170 La Celle St. Cloud, France Tel: 33 (1) 3918.12.44



YES, I'm interested in your Introductory Offer. Send more information on the Toolset and your 68000 compilers.

Send me your free brochure on The Many Facets of Quality.

Name			
Company			
Address			
City	State	Zip	
Phone			
Alsus. Inc. • 1432 Mai	n Street • Waltham, MA C	2154	DDJ 9/8

CIRCLE 273 ON READER SERVICE CARD

DDJ 9/87

DEVICE DRIVERS

(continued from page 45)

quested function; it just saves the pointer to the request header and then returns. The request header pointer segment is stored in the code segment variable req_hdr_seg, and the offset is stored in the code segment variable req_hdr_off (see the listing).

Device Interrupt Function

The device interrupt function is where the command stored in the re-

quest header is carried out. Although this function is called the interrupt function, DOS invokes it as a far call as opposed to a processor interrupt. This function has seven crucial tasks to perform, some of which replace the C compiler's start-up code. The listing of prndrv.asm has the sections marked as ;STEP 0, ;STEP 1, and so on, where:

• STEP 0 saves the machine state. On entry, the interrupt function saves the machine state by pushing all the registers onto DOS' stack, which has

enough space to have the registers pushed onto it. Then the DOS stack frame—the ss and sp registers—is saved in the code segment variables caller_ss and caller_sp.

• STEP 1 gets the driver's segment. The first step in establishing addressability is to get the driver's segment address and hold it in the ax register. Remember that the driver's segments are all in one physical segment, so when the driver is called, the value of the cs register is the segment value for all segments.

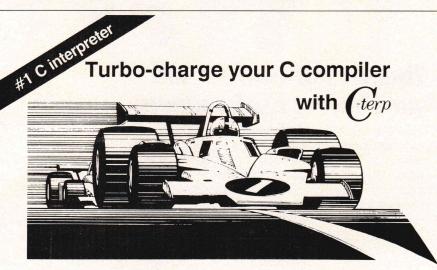
• STEP 2 makes the request header addressable. The long pointer to the request header was previously saved in the strategy function. Now the interrupt function uses this information to copy it into the private request header.

• STEP 3 finishes establishing addressability. The interrupt function moves the segment address of the driver from ax into the ds, es, and ss registers; gets the offset of the private stack and puts it into sp to establish the stack frame; and finally, sets the bp register equal to the sp register as this condition is needed to call C functions that rely heavily on bp.

• STEP 4 calls the first C function. Now the interrupt function is ready to call the first C function, which will in turn call others to perform the requested task. It calls a function named driver_functions(cmd_code) and passes the command code from the request header as a parameter. Calling a C function from assembler requires pushing the parameters onto the stack, calling the function, and removing the parameters from the stack.

• STEP 5 updates DOS' request header. The task DOS wanted the driver to perform has finished. The status of the operation is stored in the addressable copy of the request header along with any function-specific return info. In order for DOS to get this info, the request header must be copied back to the request header that DOS gave a pointer to way back in the strategy function.

• STEP 6 cleans up. To clean up, the original DOS stack frame that was saved in *caller_sp* must be restored, then the registers pushed onto DOS' stack must be popped. Finally, the interrupt function returns, and its tasks are finished.



Our C Interpreter provides the finest and fastest development environment for C and is compatible with your compiler.

- Fast Semi-Compilation -- We convert source to tokens faster than any product (existing or announced) on the market.
- Interactive Debugging -- See your code come to life as you single step, set breakpoints, call functions, view data, execute any C expression.
- Complete Language -- We've always supported full K&R, now we support the usual ANSI enhancements as well (structure assignment, enumerations, etc.) as well as keywords cdecl and far.
- Multiple Modules -- an accurate reproduction of a typical multiple module compiler environment brought to you in a high speed interactive interpreter.
- Multi-file, configurable editor -- features fast screens, inter-file copies and moves, etc. etc. Spring from file to file, module to module. Develop as you never did before. Completely reconfigurable.
- Complete Compatibility -- For each supported compiler we provided a separate C-terp with separate documentation (each compiler is a little bit different). We provide a batch file to link in your compiler's entire library. We make sure the data alignment, bit field order, and pre-processor variables are compatible with your compiler. We care about compatibility.
- Shared symbols option -- for those large 75-module applications.

- Software Paging -- for those big jobs. Our new and improved paging can now access Extended Memory directly.
- Pointer checking -- An out-of-bounds assignment will put you into debug mode with the offending statement highlighted.
- Object module support -- Link in not only your compiler's library but your own libraries (large model), assembler routines, and commercial libraries such as Essential Graphics, HALO, Windows for Data, Greenleaf, Vitamin C, etc. Our function pointers are compatible with compiled C (a must for using commercial libraries) and we support call in (from compiled to interpreted) as well.
- Numerous other features including our own batch mode, dual display and graphics support, tracing and 8087/80287 as well.

Order C-terp TODAY (Specify Compiler) Microsoft, Lattice, Aztec, C86, Mark Williams, Xenix

PRICE: MS-DOS 2.x and up - \$298 Xenix 286 System V - \$498

VISA, MC, COD * 30 Day Money Back Guarantee

* C-terp is a trademark of Gimpel Software.

EMPEL SOFTWARE

3207 Hogarth Lane * Collegeville, PA 19426 (215) 584-4261

Clarify and document your source listing and get an "organization chart" of your program's structure

with two NEW utilities from Aldebaran Laboratories, for C, BASIC, Pascal, dBASE,® FORTRAN and Modula-2 programmers, Now works

"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

- PC Magazine Sept. 16, 1986

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. Order both and save. Only \$155.00.

800-257-5773 Dept. 58

800-257-5774 Dept. 58

MasterCard, VISA, American Express, COD. Add \$5 for shipping/handling.

or see your local dealer!

Source Print and Tree Diagrammer are trademarks of Aldebaran Labs. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

Source Print

organizes your source code, simplifies debugging, and makes documentation a snap! It lists one or more

source files with informative page headings and op-

tional line numbers. while offering invaluable features:

The Index

(Cross-Reference list) saves you time by showing exactly where variables are

used and where functions, procedures, and routines are called.

Before if ((d = ares[iar][1]) == 0) p = &(ares[iar][1]); while (d = *p) 100p++;

BASIC After Wed 12-31-86 07:22:03 INDEX (Cross Ref) all identifiers 53.2293 54.2331 54.2354 53=2319 54.2336 54.2365

with **FORTRAN**

Index

Locations where new

values may be assigned to variables are shown, making it easy to track down that mysterious value change.

Structure Outlining solves the problem of hard-tosee nested control structures by automatically drawing lines around them.

Automatic Indentation of source code and listings reduces your editing time and ensures indentation accuracy.

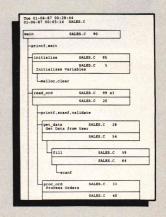
Plus . . . Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a

-06-86 13:45:44 dem3.prg **dBASE**

Tree Diagrammer

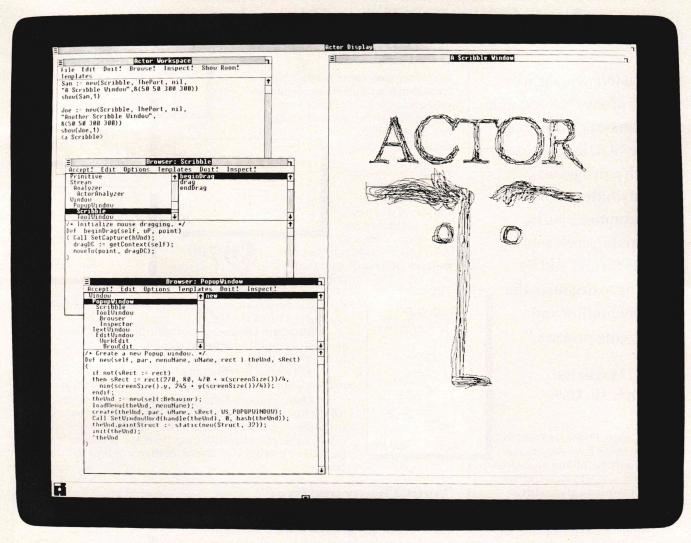
shows your program's overall organization at a glance. Ordinary program listings merely display functions, procedures, and subroutines sequentially, but do not display the relationships between these routines. Our revolutionary new Tree Diagrammer automatically creates an "organization chart" of your program showing the hierarchy of calls to functions, procedures, and subroutines. Recursive calls are indicated and designated comments in the source code will appear on the chart.

Tree Diagrammer helps you organize your program more logically. And you'll be amazed at how easy it is to debug when you see how your routines interact.



Aldebaran Laboratories 3339 Vincent Rd YES! Rush me Source Print \$\@\\$97 Both \\$155. Ship/Handling \\$5. For CA add 6\% tax _ Name	Tree Diagrammer @ \$77 Total
CompanyAddressCity	State Zip
☐ Check enclosed ☐ VISA ☐ MasterCard ☐ A Card #	merican Express Exp. Date

HOW TO WRITE AWINDOWS APPLICATION IN TEN MINUTES.



Actor™ is a new language that combines Microsoft®Windows with object-oriented programming. This means you can produce mouse and window applications very quickly.

For example, we created a simple "paint" program, and used it to draw the Actor logo you see on the screen. The whole program only took ten lines

and ten minutes. Part of it is in the middle window on the left.

Above, you see the commands that initialized the paint window and made it appear on the screen. Below, some code that's built into Actor, specifying window behavior. Through a process known as "inheritance," it's called into play automatically.

Try programming in this new way, and you'll never go back.

Find out about Actor.
Call The Whitewater Group, (312) 491-2370.

Technology Innovation Center

Technology Innovation Center 906 University Place, Evanston, IL 60201

CIRCLE 282 ON READER SERVICE CARD

DEVICE DRIVERS (continued from page 48)

Implementing a Device

Once you've completed the DOS device start-up code, you need to write the functions that actually implement the device. Each device will have its own unique requirements, and the implementor must be on intimate terms with the device to be controlled. The device presented here is extremely simple as devices go-being a printer it is a character device that has output functions only. The start-up code does not change (except for the attribute word and device name, both in the device header) with the complexity of the specific device, however.

After the start-up code is set up for calling C compiled functions, it calls driver_functions(cmd_code), passing the command code from the request header. Driver_functions() is in drvfunc.c (Listing Three, page 72). The function driver_functions() is implemented as a finite state machine that calls the device implementation functions indirectly. Function-_table is declared in drvfunc.c as an array of pointers to functions returning void, and this array is initialized to contain pointers to those functions that this device implements. Slots for unused functions are initialized to a pointer to bad_cmd(), a function that sets the request header status word to indicate an invalid command (see error.c, Listing Four, page 74). The command code is the state that selects which function to call.

For those not familiar with an array of pointers to functions, here's an explanation: Each element in the array is a pointer to, or the address of, a function that (for an 8086 small model) is an offset into the code segment. The index into the array determines which address to call, just as the index into an array of integers determines which integer to access. If you have the address of a function, you can call it indirectly as (void)(* function_pointer)(); and by extending this to an array, you get (void)(* function_pointer_array[index])();

There's nothing really esoteric about it—assembly-language programmers call it a jump table. This construct produces efficient code, which is what you want in a driver.

Changing Your Address?

To change your address, attach your address label from the cover of the magazine to this coupon and indicate your new address below.

Name	
Address	
Address	Apt. #
City	State
Zip	

Mail to: Dr. Dobb's Journal, PO Box 27809, San Diego, CA 92128

JYRCC FORMAKER

A POWERFUL SCREEN AND WINDOW MANAGER

UNIX[™] XENIX[™] MS-DOS[™] PRIMOS[™]

JYACC'S FORMAKER makes it easy to design, develop, test and document interactive applications. FORMAKER includes a utility for creating and maintaining forms and windows and a subroutine library to provide access to them.

- Reduced Development Time
- Less Complex Programs
- Advanced User Interface
- Easy Form Creation
- System Prototyping
- Self-Documenting
- Application Portability
- Display Forms
- Windows Management
- Edits and Validations
- Display Prompts
- Error Messages
- Save Data
- Restore Data
 - Cursor Control
- Pop-Up Windows

■ Free-Form Design

■ Easy-To-Use

"'Test Mode''

■ Interactive Form Editing

JYACC, INC. is a project-oriented computer consulting firm providing services in most areas of system design and implementation. Call us today to discuss your specific needs and applications.

Available for the IBM PC/XT/AT and compatibles, AT&T 7300 and 3B family, DEC Vax and Micro VAX, NCR Tower, Prime 50 series, Altos 986, Fortune 32:16, Gould Concept series, HP 9000 and TI PC family.

UNIX is a trademark of AT&T XENIX and MS-DOS are trademarks of Microsoft PRIMOS is a trademark of Prime Computer



JYRCC INC.

116 John Street New York, New York 10038 212-267-7722 Outside NY call 1-800-458-3313

DEVICE DRIVERS (continued from page 51)

The alternative is a switch statement or a series of *if...else* pairs that compile into a lot of compare/jump instructions. Note that all the functions to which pointers are placed in the array must take the same number of parameters or chaos will result. In C, a function name (without the parentheses) evaluates to a pointer to that function.

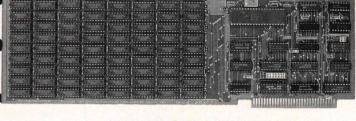
All DOS device drivers must implement an initialization functioncalled only once when the driver is first loaded. Mine is called init() and is found in init.c (Listing Five, page 74). The initialization function for a character device must return to DOS a long (segment:offset) pointer to the end of the driver and set the done bit in the status word. The ending address of the driver is the segment address obtained by a call to show_cs() (in show_cs.asm, Listing Six, page 75) and a short pointer (offset only) to an unsigned integer called _Uend. _Uend is inserted by the Aztec linker, ln, at the end of the uninitialized data segment. Those using other compilers/linkers need to replace this by creating another module called end.c or whatever with a glob-

All device drivers must include an initialization function.

al unsigned called _Uend and make this the last module in the list when you link. My driver also displays a message on the screen, resets the parallel port, and sets the printer font to elite (12 characters/inch). For a different device, the initialization will be quite different. If this were a block device driver, in addition to all this, the initialization function would have to set the number of units and return a long pointer to an array of BPBs (BIOS parameter blocks).

Being a printer, the primary function this device performs is output(), found in output.c (Listing Seven, page 75). Here you get from the request header the number of bytes to transfer and the segment:offset at which you find the first byte. For each character, the output function gets the character, sends it to the parallel port by calling char_2_prn()(in prlport.asm, Listing Eight, page 77), checking for an error return status, and finally incrementing the transfer offset and character transferred count. If an error condition is returned by char_2_prn(), the error handler function error() is called; otherwise, when you have finished. you set the done bit in the request header status word. Again, this is a simple device, but you apply the same technique to develop drivers for more complex devices.

SemiDisk® has an attractive personality.



"A while back I got a SemiDisk to help me with my database work. A SemiDisk is like a RAMdisk only a whole lot better. It doesn't sit in my main or EMS memory, and, using the Battery Backup, it's like permanent storage.

"That SemiDisk makes light work of the jobs that were sending my hard disk to an early grave. And SemiDisk has no head to crash; no moving parts to wear out. With all the time it saves me, I figure it paid for itself in just a couple of months.

"Then I heard programs like Microsoft Windows could use my SemiDisk for temporary files instead of using EMS. So I moved them

over to the SemiDisk, too. The quiet speed of it is almost elegant!

"My boss wanted to try my SemiDisk on the company LAN server, but I told him to get his own. A couple of days later, he was wearing a grin as big as mine. I guess he likes his SemiDisk too.

"One morning I booted up my computer and there was my word processor waiting for me on the SemiDisk. I swear I didn't put it there! After I tried it, I knew it was there to stay.

"Meanwhile, I've found a new use for my hard disk, too. It's <u>great</u> for backing up my SemiDisk!"

I/O mapped SemiDisk goes in standard PC, XT or AT expansion slot. Priced at just \$495 for 512K, \$795 for 2Mb. Battery Backup \$130. Up to 8Mb per drive. Call or write for further information or to place an order.

SemiDisk

End the waiting.

SemiDisk Systems, Inc. P.O. Box GG Beaverton, OR 97075 (503) 626-3104





CIRCLE 85 ON READER SERVICE CARD

Return Info

Drivers return status information to DOS via the status word in the request header. This is represented by an unsigned integer called status in the request header structure reg_hdr (see rh.h). Bit 15 is the error bit, and it is set to 1 if the driver needs to report an error condition to DOS, with the 8 least significant bits holding the error code. Code to implement an error condition is in error.c, and for my printer driver, the only possible error codes are those that are returned by the parallel port BIOS. This function is called only when errors occur; otherwise, the driver functions set bit 8, the done bit, to 1 indicating to DOS that the device has completed its task successfully.

Linking

Linking the driver is different from linking a normal C program. You must make all the logical segments fall into the same physical segment, and the device header must be at offset 0 in the file. One of the reasons why I use Aztec C is because its linker, ln, can accomplish all the preceding tasks in response to a few command-line arguments, making it a useful and powerful tool for dealing with the 8086 architecture.

To link the driver using ln, you type:

ln -t -b 0 -c 0 -o prndrv.com prndrv.o [list of files and libraries]

where -t saves the symbol table in prndev.sym, -b 0 sets the base address to 0h (same as ORG 0), -c 0 makes the code segment start at offset 0h in the physical segment, and -o prndrv.com means that the output file is prndrv.com (specifying an extension of .COM causes all logical segments to be in one physical segment).

Again, users of other compilers/ linkers will have to adapt this to their systems. Hint: Try using a GROUP statement in prndrv.asm to cause the code and data segments to be joined into one physical segment, and remember to take the offset into the group instead of into the segment when using the offset operator.

Known Shortcomings

All variables declared outside a function must be initialized to some value-they cannot be assumed to be initialized to 0, and if they are not initialized, the scheme gets buggy. Drivers written in C tend to be largeprobably all those references to bp.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listings begin on page 68.)

Vote for your favorite feature/article. Circle Reader Service No. 4.

Get real productive with REAL-TOOLS, a general purpose set of "C" development tools for UNIX™ and XENIX™

Get Graphics Too! In addition to an advanced screen management system and superior windowing capabilities, REAL-TOOLS offers user-defined graphics for you to draw, save, recall, copy and animate symbols and panels.

So if you're developing applications for the real world - get real productive. Get graphics. Get REAL-TOOLS.



\$99 Binary only. \$549 Library source. \$999 Complete source.

Pioneering Controls Technologies, Inc. 510 Bering Drive, Suite 300, Houston, Texas 77057 (713) 266-8649

™REAL-TOOLS is a trademark of Pioneering Controls Technologies, Inc.
™UNIX is a trademark of A1&T.
™XENIX is a trademark of Microsoft Corporation.

CIRCLE 191 ON READER SERVICE CARD

MAMMOTH PROJECTS OFTEN FAIL WITHOUT THE RIGHT TOOLS.



WITHOUT THE PROPER TOOLS, ANY COBOL APPLICATION CAN BE A MAMMOTH PROJECT.

> Generate native COBOL screen handling source code for your application.

Or, use COBOL spll's powerful runtime facility.

With COBOL spll, you make the choice! We don't make it for you.

Create interactive demos, tutorials and application prototypes with COBOL spll's dialogue facility.

COBOL spll's powerful panel painter automatically attributes, generates automatic borders and lets you move or copy blocks of the panel within or across panels.

Practically all of your field editing can be done with COBOL spll. Perform range and discrete value checking as well as binary value checking

30 Day Money Back Guarantee!

Only \$345.00! After August 31, 1987 the normal retail price of \$395.00 will go into effect.

Give us a call and we'll send you our free demo. We think you'll be impressed with the power and flexibility of COBOL spll.

COBOL spll supports RM COBOL, Realia COBOL, Microsoft COBOL and RM COBOL 8X.

COBOL SPIL . . . THE MISSING LINK TO COBOL PRODUCTIVITY!

Flexus International Corporation P.O. Box 9119 Morristown, NJ 07869 (201) 895-4724

CIRCLE 189 ON READER SERVICE CARD

FILE COMPARISONS

```
Listing One (Text begins on page 28.)
** Copyright (c) 1987, Tom Steppe. All rights reserved.
** This module compares two arrays of lines (representing
** files) and reports the sequences of consecutive matching
** lines between them using the "recursive longest matching
** sequence" algorithm. This is useful for implementing a
** file comparison utility.
** Compiler: Microsoft (R) C Compiler Version 4.00
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <malloc.h>
/* Boolean type and values. */
typedef int
              BOOLEAN;
#define TRUE
#define FALSE 0
/* Minimum macro. */
#define min(x, y) (((x) <= (y)) ? (x) : (y))
/* Value to indicate identical strings with strcmp. */
#define ALIKE
/* Result of hashing function for a line of text. */
typedef unsigned int
                      HASH;
/* Mask for number of bits in hash code. (12 bits). */
#define MASK (unsigned int) 0x0FFF
/* Number of possible hash codes. */
#define HASHSIZ (MASK + 1)
/* Information about an entry in a hash table. */
typedef struct tblentry
    int frst; /* First line # with this hash code. */
int last; /* Last line # with this hash code. */
                 /* First line # with this hash code. */
} TBLENTRY;
/* Information about a line of text. */
typedef struct lineinf
    HASH hash; /* Hash code value. */
int nxtln; /* Next line with same hash (or 0). */
} LINEINF;
/* Information about a file. */
typedef struct fileinf
              **txt;
                          /* Array of lines of text. */
              *line;
    LINEINE
                          /* Array of line info structs. */
    TBLENTRY *hashtbl;
                         /* Hash table. */
} FILEINF;
/* Function declarations. */
BOOLEAN filcmp (char **, int, char **, int, int);
                   (char **, int, FILEINF *);
BOOLEAN get_inf
HASH
       calc_hash (char *);
      BOOLEAN chk_hashes (LINEINF *, LINEINF *, int);
     cnt_matches (char **, char **, int);
int
void
       rpt seq
                   (int, int, int);
/******************************
** compare compares two arrays of lines and reports the
** sequences of consecutive matching lines. The zeroth
```

```
** element of each array is unused so that the index into
** the array is identical to the associated line number.
** RETURNS: TRUE if comparison succeeded.
           FALSE if not enough memory.
BOOLEAN compare (al, nl, a2, n2, lngval)
char
      ** 1 :
                /* (I) Array of lines of text in #1. */
int
      n1:
                /* (I) Number of lines in al.
                       (Does not count 0th element.) */
               /* (I) Array of lines of text in #2. */
      **22:
char
              /* (I) Number of lines in a2.
int
      n2;
                       (Does not count 0th element.) */
      lngval; /* (I) "Long enough" value. */
int
            f1;
                   /* File information for #1. */
   FILETNE
           f2;
                   /* File information for #2. */
   FILEINF
            rtn; /* Return value. */
   BOOLEAN
   /* Gather information for each file, then compare. */
       (get inf (al, nl, &fl) && get inf (a2, n2, &f2)))
       fnd seq (&f1, 1, n1, &f2, 1, n2, lngval);
   return (rtn);
/********************
** get inf calculates hash codes and builds a hash table.
** RETURNS: TRUE if get inf succeeded.
           FALSE if not enough memory.
static BOOLEAN get inf (a, n, f)
         **a;
               /* (I) Array of lines of text. */
int
         n;
               /* (I) Number of lines in a. */
        *f:
              /* (O) File information. */
FILEINE
                           /* Size of hash table. */
    unsigned int size;
    register int i;
                           /* Counter. */
                  *entry; /* Entry in hash table. */
    TRIENTRY
    /* Assign the array of text. */
    f \rightarrow txt = a;
    /* Allocate and initialize a hash table. */
    size = HASHSIZ * sizeof (TBLENTRY);
    if (f->hashtbl = (TBLENTRY *) malloc (size))
       memset ((char *) f->hashtbl, '\0', size);
    }
    else
       return (FALSE);
    /* If there are any lines: */
    if (n > 0)
        /* Allocate an array of line structures. */
       if (f->line = (LINEINF *)
               malloc ((n + 1) * sizeof (LINEINF *)))
            /* Loop through the lines. */
           for (i = 1; i <= n; i++)
```

(continued on next page)

SEIDL VERSION MANAGER

RECONSTRUCT any VERSION of a SOFTWARE product AUTOMATICALLY.

- ★ Archive Database tracks all source code revisions as well as annotative comments.
- ★ Audit Trail Report

 provides user specified information on any aspect of a project's development.
- ★ Revision Branching allows any number of revisions to be created from an existing revision.
- ★ Text Compression
 optionally reduces disk storage
 requirements.
- ★ Menu Driven Shell makes SVM easy to use.
- ★ Price: \$299.95 + \$5.00 p&h.

SEIDL MAKE UTILITY

NOT just ANOTHER COPY of the UNIX MAKE.

- ★ Structured Language to describe dependencies in a clear, concise and portable manner.
- ★ Rich Command Set includes parameterized macros, variables, if-then-else, iteration, wild cards, macro libraries, interactive statements, environment access and much more!
- ★ Intelligent Analysis
 algorithm handles nested include
 files, library dependencies, and
 performs consistency tests to
 detect errors that other makes
 would blindly ignore.
- *** Price:** \$99.95 + \$3.50 p&h.

CALL TODAY

1-313-662-8086

Visa/MC/COD Accepted Dealer Inquiries Invited

SEIDL COMPUTER ENGINEERING

3106 Hilltop Dr., Ann Arbor, MI 48103

FILE COMPARISONS

Listing One (Listing continued, text begins on page 28.)

```
/* Calculate the hash code value. */
              f->line[i].hash = calc hash (f->txt[i]);
              /* Locate the entry in the hash table. */
              entry = f->hashtbl + f->line[i].hash;
              /* Update the linked list of lines with */
              /* the same hash code.
              f->line[entry->last].nxtln = i;
              f->line[i].nxtln = 0;
              /* Update the first and last line */
              /* information in the hash table. */
              if (entry->frst == 0)
                  entry->frst = i;
              entry->last = i;
       }
       else
       {
          return (FALSE);
   else
      f->line = NULL;
   return (TRUE);
/**********************************
** calc_hash calculates a hash code for a line of text.
** RETURNS: a hash code value.
static HASH calc hash (buf)
char *buf; /* (I) Line of text. */
   register unsigned int
                         chksum; /* Checksum. */
                                  /* Pointer. */
                         *s:
   HASH
                         hash;
                                  /* Hash code value. */
    /* Build up a checksum of the characters in the text. */
   for (chksum = 0, s = buf; *s; chksum ^= *s++)
   /* Combine the 7-bit checksum and as much of the */
   /* length as is possible.
   hash = ((chksum & 0x7F) | ((s - buf) << 7)) & MASK;
   return (hash);
/*******************
** Given starting and ending line numbers, fnd_seq finds a
** "good sequence" of lines within those ranges. fnd seq
** then recursively finds "good sequences" in the sections
** of lines above the "good sequence" and below it.
static void fnd_seq (f1, beg1, end1, f2, beg2, end2, lngval)
FILEINF
         *f1:
                 /* (I) File information for #1. */
int
         beg1;
                 /* (I) First line # to compare in #1. */
int
                 /* (I) Last line # to compare in #1. */
        end1:
```

```
FILEINE
          *f2:
                    /* (I) File information for #2. */
                    /* (I) First line # to compare in #2. */
int
          bea2:
                   /* (I) Last line # to compare in #2. */
int
          end2;
          lngval; /* (I) "Long enough" value. */
int
    T.TNE.TNE
                 *linel;
                            /* Line information ptr in #1. */
    LINEINF
                 *line2;
                           /* Line information ptr in #2. */
                            /* Looping limit. */
    register int limit;
    int
                 ln1;
                            /* Line number in #1. */
    int
                 ln2;
                            /* Line number in #2. */
    register int ln;
                           /* Working line number. */
    BOOLEAN
                           /* Continue to loop? */
                 qo;
                           /* Longest possible seq. */
    int
                 most:
                           /* Longest possible due to #1. */
    int
                 most1;
                           /* Longest possible due to #2. */
    int
                 most2:
                           /* Length of longest seq. */
                 cnt:
                           /* Length of prev longest seq. */
    int
                 oldcnt;
                            /* Length of cur longest seq. */
    int
                 n;
                            /* Line of longest seq. in #1. */
    int
                 m1;
                           /* Line of longest seq. in #2. */
    int
                 m2;
    /* Initialize. */
    go = TRUE;
    line1 = f1->line;
    line2 = f2->line;
    /* Initialize longest sequence information. */
    cnt = 0;
                        /* Length of longest seq. */
                         /* Line # of longest seq. in #1. */
           = beg1 - 1;
    m2
           = beg2 - 1;
                        /* Line # of longest seq. in #2. */
    oldcnt = 0;
                          /* Length of prev longest seq. */
    /* Calculate maximum possible number of consecutive */
    /* lines that can match (based on line # ranges). */
    most1 = end1 - beg1 + 1;

most2 = end2 - beg2 + 1;
    /* Scan lines looking for a "good sequence".
    ** Compare lines in the following order of line numbers:
    **
    **
                        (1, 1)
    ** (1, 2), (2, 1), (2, 2)
** (1, 3), (2, 3), (3, 1), (3, 2), (3, 3)
    ** etc.
    for (ln1 = beg1, ln2 = beg2; TRUE; ln1++, ln2++)
        if (ln2 \le end2 - cnt)
        /* There are enough lines left in #2 such that it */
        /* is possible to find a longer sequence.
             /* Determine the limit in #1 that both
             /* enforces the order scheme and still makes */
             /* it possible to find a longer sequence.
             limit = min (ln1 - 1, end1 - cnt);
             /* Calculate first potential match in #1. */
             for (ln = f1->hashtbl[line2[ln2].hash].frst;
                     ln && ln < beg1; ln = line1[ln].nxtln)</pre>
             }
             /* Loop through the lines in #1. */
             for (; ln && ln <= limit; ln = line1[ln].nxtln)
                 if (line1[ln].hash == line2[ln2].hash &&
                         line1[ln + cnt].hash ==
                             line2[ln2 + cnt].hash &&
                          !(ln - m1 == ln2 - m2 \&\&
                         ln < m1 + cnt && m1 != beg1 - 1))
                 /* A candidate for a longer sequence has */
```

(continued on next page)



FULL TEXT RETRIEVAL

FIND

Searches 5,000 text files in 5 seconds.

The ultimate personal computer information retrieval software.

Find any information created by the popular word processors or ASCII files, in seconds-without having to set up a database or do programming. Scan a 20 Mbyte hard disk or a collection of floppies and instantly display all occurrences of a name, a product, a phrase, a number, or anything else you need to find. Save hours of manual searching.

TEXT

FEATURES

- Comprehensive searches create a search request with any combination of AND, OR, NOT, and WITHIN (WITHIN refers to how far apart two words or phrases can be — up to 30,000 words).
- Wild Cards, for example: type "micro*" to get (microcomputer, microcomputing, micro-processor, etc.).
 Mark and Save retrieved information to create a new file.
- Highlights search-words and phrases in retrieved text.
 Find Function automatically displays search topic in the retrieved file.
 Phrase Search, in addition to single word search.

SYSTEM REQUIREMENTS MINIMUM

• 384K • Two Disk Drives • DOS 2.0 or above • Designed for IBM PC, XT, AT, compatibles, and most MS-DOS computers

FAST

ZyINDEX Personal \$95

Searches up to 325 files

ZyINDEX Standard \$145 Searches up to 500 files

ZyINDEX Professional \$295

Searches up to 5,000 files

ZVINDEX Plus \$695

Searches up to 15,000 files with LAN capabilities.

30 Day Money Back Guarantee

ZyLAB TA

233 East Erie Street Chicago, Illinois 60611 (312) 642-2201 (800) 544-6339 For orders and information

CIRCLE 329 ON READER SERVICE CARD

FILE COMPARISONS

Listing One (Listing continued, text begins on page 28.) /* been located. The current lines /* match, the current lines + cnt match, */ /* and this sequence is not a subset of */ /* the longest sequence so far. /* Calculate most possible matches. */ most = min (end1 - ln + 1, most2);/* First compare hash codes. If the */ /* number of matches exceeds the /* longest sequence so far, then /* compare the actual text. if (chk_hashes (line1 + ln, line2 + ln2, cnt) && (n = cnt matches (f1->txt + ln, $f2\rightarrow txt + ln2, most)) > cnt)$ /* This is the longest seq. so far. */ /* Update longest sequence info. */ oldcnt = cnt; cnt = n; m1 = ln; m2 = ln2; /* If it's long enough, end the */ /* search. if (cnt >= lngval) { break; /* Update limit, using new count. */ limit = min (ln1 - 1, end1 - cnt); } } } /* If it's long enough, end the search. */ if (cnt >= lngval) break; } most2--; } else { go = FALSE; /* This file is exhausted. */ /* Repeat the process for the other file. */ if (ln1 <= end1 - cnt) limit = min (ln2, end2 - cnt); for (ln = f2->hashtbl[line1[ln1].hash].frst; ln && ln < beg2; ln = line2[ln].nxtln)</pre> } for (; ln && ln <= limit; ln = line2[ln].nxtln) if (line1[ln1].hash == line2[ln].hash && line1[ln1 + cnt].hash == line2[ln+ cnt].hash && !(ln1 - m1 == ln - m2 &&ln1 < m1 + cnt && m2 != beg2 - 1)) most = min (end2 - ln + 1, most1);if (chk_hashes (line1 + ln1, line2 + ln, cnt) &&

```
(n = cnt_matches (f1->txt + ln1,
                                f2 \rightarrow txt + ln, most)) > cnt)
                        oldcnt = cnt;
                        cnt
                               = n:
                               = ln1;
                        m2
                               = ln:
                        if (cnt >= lngval)
                        {
                            break;
                        limit = min (ln2, end2 - cnt);
                    }
                }
           }
            if (cnt >= lngval)
               break;
           most.1--:
        else if (!go)
           break; /* This file is exhausted, also. */
   }
    /* If the longest sequence is shorter than the "long */
    /* enough" value, the "long enough" value can be
    /* adjusted for the rest of the comparison process.
   if (cnt < lngval)
        lngval = cnt;
   if (cnt >= 1)
   /* Longest sequence exceeds minimum necessary size. */
        if (m1 != beg1 && m2 != beg2 && oldcnt > 0)
        /* There is still something worth comparing */
        /* previous to the sequence.
            /* Use knowledge of the previous longest seq. */
           fnd_seq (f1, beg1, m1 - 1,
                    f2, beg2, m2 - 1, oldcnt);
       /* Report the sequence. */
       rpt seq (m1, m2, cnt);
        if (m1 + cnt - 1 != end1 && m2 + cnt - 1 != end2)
        /* There is still something worth comparing */
        /* subsequent to the sequence.
            fnd seq (f1, m1 + cnt, end1,
                    f2, m2 + cnt, end2, lngval);
   }
/************************************
** chk_hashes determines whether this sequence of matching
** hash codes is longer than cnt. It knows that the first
** pair of hash codes is guaranteed to match.
** RETURNS: TRUE if this sequence is longer than cnt.
            FALSE if this sequence is not longer than cnt.
                                           (continued on next page)
```

DAN BRICKLIN'S BRICKLIN'S BRICKLIN'S DEMO PROGRAM PROGRAM PROGRAM ONLY \$74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."

—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."

-Soft · letter

Product of the Month

-PC Tech Journal

Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

NEW TUTORIAL! TUTORIAL! JUST \$49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.



Use 800-number for orders only. Questions, special shipping, etc., call **617-332-2240.** No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add \$15.00. Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the



SOFTWARE GARDEN, INC.

Dept. D P.O. Box 373, Newton Highlands, MA 02161 CIRCLE 314 ON READER SERVICE CARD

FILE COMPARISONS

Listing One (Listing continued, text begins on page 28.)

```
static BOOLEAN chk_hashes (line1, line2, cnt)
               *line1; /* (I) Line information for #1. */
*line2; /* (I) Line information for #2. */
LINEINF
LINEINE
                        /* (I) Count to try to exceed. */
   register int n;
                         /* Count of consecutive matches. */
    for (n = 1; n <= cnt &&
        ((++line1)->hash == (++line2)->hash); n++)
   return (n > cnt);
/**********************************
** cnt matches counts the number of consecutive matching
** lines of text.
**
** RETURNS: number of consecutive matching lines.
***********************************
static int cnt matches (s1, s2, most)
char **s1; /* (I) Starting line in file #1. */
char **s2; /* (I) Starting line in file #2. */
register int most; /* (I) Most matching lines possible. */
    register int n; /* Count of consecutive matches. */
    /* Count the consecutive matches. */
    for (n = 0; n < most && strcmp (*s1++, *s2++) == ALIKE;
           n++)
   return (n);
** rpt seq reports a matching sequence of lines.
static void rpt_seq (m1, m2, cnt)
     ml; /* (I) Location of matching sequence in #1. */
    m2; /* (I) Location of matching sequence in #2. */
int
int
     cnt; /* (I) Number of lines in matching sequence. */
    fprintf (stdout,
        "Matched %5d lines: (%5d - %5d) and (%5d - %5d) \n",
        cnt, m1, m1 + cnt - 1, m2, m2 + cnt - 1);
```

End Listing

It's good Vitamin C for your system!

"If you need source code, make sure your wallet is wide open or get

Picking the best value package is hard... If you're a source code fanatic like me, VITAMIN C is preferable." - Computer Language, June, 1987

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination software professionals demand to produce vital combination sortware professionals demand to produce Highly efficient, superior applications in dramatically less time.

superior applications in gramatically less time. riignly efficiency professionally crafted C code provides lightning fast displays nred by roday's window intensive programs.

High level functions provide maximum productivity and required by today's window intensive programs.

require little supporting code. Extended versions of these routines and Hexible control over specific details when necessary. Plus, Vitamin C's versatile, open ended design is full necessary. Plus, ynamin s versame, open ended design is of hooks so you can intercept and plug-in special handlers to

VCScreen, our screen painter / code generator speeds your customize or add features to most routines. development even more! Simply draw your input forms using development even more! Simply uraw your input forms using our interactive design editor and generate perfect C source code

ready to compile and link with the Vitamin C library.

ORDER NOW! (214)416-6447

Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering. VCScreen \$99.95 For IBM PC .XT, AT, PS/2 and true compatibles. Requires Vitamin C. We ship UPS. Please include \$3 for ground, \$6 for 2-day air, \$20 for ground, so for 2-day an, s20 for overnight, or \$30 if outside the U.S. overnight, or 550 it outside the U.S. All Texas residents add 7 1/4% sales tax. All

- ✓ Professional C function library
- **✓** 30 day money back guarantee
- ✓ Multiple bullet proof windows
- ✓ Easy full screen data entry
- ✓ Unlimited data validation
- ✓ Context sensitive help manager
- Menus like Lotus and Mac
- ✓ Programmable keyboard handler
- Text editor routines
- No royalties or runtime fees
- ✓ Library source included FREE
- Free technical support
- Free BBS at (214)418-0059
- ✓ Supports all major compilers including Microsoft 5.0
- ✓ VCScreen code generator too!
- ✓ UNIX version avaialable, call for details

Windows • Data Entry • Menus • Help • Text Editing

Texas residents add / 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

Finally

The New BASICs!

Programming Techniques and Library Development

by Namir Clement Shammas

Here's an in-depth look at the latest and fastest BASICs: Quick-BASIC 3.0, Turbo BASIC 1.0 and True BASIC 2.0.

The New BASICs will quickly orient programmers to the syntax and programming features of these new, improved BASICs. Now, you can learn the details of implementing subroutines, functions, and libraries to permit more structured coding.

The book includes a discussion of the new BASICs and their environments, and a look at the new programming framework, including BASIC options and data types, strings, arrays, decision making, loops, exit statements, error handling, user-defined functions, and callable subroutines.

Y ou'll also find a collection of useful programming examples and ready-to-use libraries, including libraries for general utilities, extended string management, numerical analysis, statistics, data structures, files manipulation, and sorting and searching. In addition, there's a binary-tree library, an MS-DOS files library, a library for Pascal-like sets, and much more!

 ${f B}$ est of all, all programs and subroutines are also available on disk, with full source code. MS-DOS format.

he New BASICs

Book and Disk (MS-DOS)
Book only

Item #43-7
Item #37-2

Item #43-7 \$39.95 Item #37-2 \$24.95

TO ORDER: Return this order form with your

payment to: M&T Books 501 Galveston Dr.

Redwood City, CA 94063

Redwood City, CA 94063

Or, call TOLL-FREE **800-533-4372** Mon–Fri 8AM–5PM (In CA call **800-356-2002**)

ORDER FORM

YES! Please send me **The New BASICs** book and disk, \$39.95 ____ Please send me **The New BASICs** book, \$24.95 ____

ν **Εποτεί** 500κ, φ2 1.55 _____

Subtotal _____

CA residents add sales tax ____ % _____ Add \$2.25 per item for shipping _____

Total

TOTAL ____

Charge my _____ VISA _____ M/C ____ Amer. Exp.

Card No. _____ Exp. ____

CODE: 3131C

XOR CHAIN

```
Listing One (Text begins on page 36.)
/* Binary tree delete procedure
 * Input parameter "s" points to a scan for an XOR
   chained binary tree.
 * The procedure deletes s->child from the tree,
    and returns with s set by GoParent(s)
void Delete(s)
struct Scan *s;
    struct Scan temp, *t;
   struct Item *i, *j, *k;
    i = s->child; /*i is the item to be deleted*/
   GoParent(s);
    if (i->llink == i->rlink)
                                       /* Case 1 */
         * adjust the pointers for s->child,
        * i's parent
        if (i->key < s->child->key)
           s->child->llink = s->parent;
            s->child->rlink = s->parent;
    else if (i->llink == s->child)
                                     /* Case 2 */
        * adjust the pointers for s->child,
        * i's parent
        if (i->key < s->child->key)
          s->child->llink = i->rlink ^
                             s->parent ^ s->child;
            s->child->rlink = i->rlink ^
                              s->parent ^ s->child;
         * adjust the pointers for i's child
        j = i->rlink ^ s->child;
        j->rlink ^= i ^ s->child;
j->llink ^= i ^ s->child;
   else if (i->rlink == s->child)
                                      /* Case 3 */
        * adjust the pointers for s->child,
        * i's parent
        if( i->key < s->child->key )
         s->child->llink = i->llink ^
                             s->parent ^
                              s->child;
           s->child->rlink = i->llink ^
                              s->parent ^
                              s->child:
        * adjust the pointers for i's child
       j = i->llink ^ s->child;
        j->rlink ^= i ^ s->child;
                               (continued on page 65)
```

Create 68K Embedded Systems On ATs

Oregon Software Brings VAX and VME Pascal-2® Cross-Development Tools To MS-DOS Workstations

Get All Your Tools In One Package

The Pascal-2 Cross-Development System gives you price/performance value that beats high-priced work-stations. Compare these features and call for further information.

Compiler

- 68000, 68020, and 68881 instruction sets
- ROMable code
- reentrant code
- separate program sections for code and data
- easy access to hardware through fixed memory locations (I/O space)

Assembler-Linker

- VERSAdos compatible output
- easily linked, relocatable S-record format

Concurrent Program Support

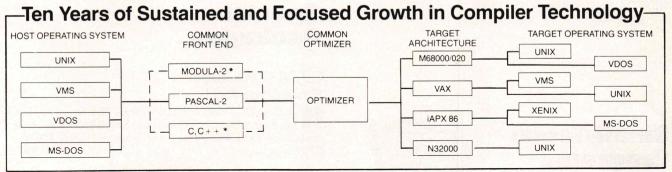
- process synchronization primitives for creating a multiprocessing real-time kernel
- interactive post-mortem analyzer
- software-selectable interrupt levels

Stand-Alone Library

- source supplied: users may adapt code as needed
- user programs that run without an operating system or other real-time kernel
- compact code for extremely small ROM programs or limited memory devices
- ability to write device drivers in Pascal or assembly code
- user-customized system services or control programs
- easily-gathered performance analysis statistics

Source Management

- a version-control system to track and identify each phase of development
- optional branching to multiple versions from a single source
- automated rebuilding of programs from component modules (a Unix-style Make function)
- a restricted-access system for coordinating revisions



*Release planned in 1988.



800-367-2202

6915 SW Macadam Avenue, Portland, Oregon 97219 U.S.A. 503-245-2202 FAX 503-245-8449 TWX 910-464-4779

DISTRIBUTORS: ■ DENMARK erik mainz a/s, Theklavej 46, DK-2400 Copenhagen NV, Tel: 1-34-7788 ■ ENGLAND Real Time Products, 1 Paul Street, London EC2A 4JJ, Tel: 1-588-0667; Grey Matter, Microcomputer Software and Consultancy, 4, Prigg Meadow, Ashburton, Devon TQ13 7DF, Tel: 364-53499 ■ FRANCE Focal Informatique, 5 Place de la Gare, Le Rhodanien niveau 7, 69003 Lyon Part-Dieu, Tel: 72-33-02-02; SCT Electronique, Za Route du Bua, Batiment B-Entree 2, Cidex 434 Verrieres Le Buisson, Tel: 1-6011 1950 ■ THE NETHERLANDS Computing & Systems Consultants BV, Stationsplein 47, 5611 BC Eindhoven, 040434957 ■ SWEDEN IND AB, Box 2066, S-175 02 Jaerfalla, Tel: 75852025 ■ SWITZERLAND Fabrimex AG Kirchenweg 5, 8032 Zurich, Tel: 01-2512929; Muhlethaler AG Computer Systems Kirchstrasse 2, CH-4536 Attiswil, Tel: 65772911 ■ WEST GERMANY AC Copy Kurbrunnenstrasse 30, Aachen D-5100, Tel: 241-505477.

The following are trademarks: Pascal-2, SourceTools, and Oregon Software, Oregon Software, Inc., VAX and VMS Digital Equipment Corporation, UNIX, AT&T Bell Laboratories, Incorporated. MS, and MS-DOS are registered trademarks of Microsoft Corporation.

Small-C Handbook & Small-C Compiler



his compiler and handbook provide everything you need for learning how compilers are constructed, and for learning C at its most fundamental level. You'll find a discussion of assembly language concepts and program translation tools, and of how to generate a new version of the compiler. Full source code is included. Please specify MS/PC-DOS or CP/M: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

CP/M Compiler & Handbook Item #006B \$37.90 MS/PC-DOS Compiler & Handbook Item #006C \$42.90

Small-Tools: Programs for Text **Processing**



his package of Small C programs performs specific, modular operations on text files. It is supplied as source code with full documentation. With the Small-C Compiler. you can select and adapt these tools to meet your needs. Please specify MS/PC-DOS or CP/M: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD. Small-Tools

Item #010A \$29.95

Small-Mac: An Assembler for Small-C



his package includes: a simplified macro facility, C his package includes, a simplified language expression operators, object file visibility, descriptive error message and an externally defined instruction table. Source code and documentation is included. CP/M only. Please specify: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

Small-Mac Item #012A \$29.95

Special **Packages**

CP/M C Package Save Over \$27!

eceive: Dr. Dobb's Toobook of C, The Small-C Handbook and Small-C Compiler, Small-Mac Assembler, and Small-Tools Text Processing Programs. Only \$99.95! CP/M Package Item #005A \$99.95

MS/PC-DOS C Package Save \$22

eceive: Dr. Dobb's Toolbook of C, The Small-C Handbook and MS/PC-DOS Addendum, Small-C Compiler, Small-Tools Texts Processing Programs and Small Windows. Only \$109.95!

Specify Microsoft C Version 4.0, Small C or Lattice C com-

MS/PC-DOS Package

Item #005W

\$109.95

C Disk Formats

Please indicate MS/PC-DOS or CP/M. For CP/M specify: Apple, Osborne, Kaypro, Zenith Z-100 DS/ DD, 8" SS/SD.

For Small-Windows, specify Small-C, Microsoft C Version 4.0 or Lattice C compiler.

C Chest and Other C Treasures from Dr. Dobb's Journal

his comprehensive anthology contains the popular "C Chest" columns from *Dr. Dobb's*, along with the lively philosophical and practical discussions they inspired. You'll also find other information-packed articles by C experts.

C Chest and

Other C

Treasures

Topics covered include: pipes, wild-card expansion, and quoted arguments; sorting routines; command-line processing; queues and bit maps; utilities such as *ls, make*, and *more*; expression parsing; hypenation; IBM cursor control and an Fget that edits; redirection; accessing IBM video display memory; trees; an AVL tree database package; directory traversal; sets; shrinking. EXE file images; hashing, expressions, and Roman numerals; and statistical applications of digital low-pass filters.

Other C treasures include: a variable metric minimizer; Christensen protocols; Fgrep; a peephole optimizer; curve fitting with cubic splines; and the CompuServe B protocol.

All subroutines and programs are written in C and are available on disk with full source code. MS-DOS format.

Book & MS-DOS Disk Item #49-6 \$39.95

Book only Item #40-2 \$24.95

NOW FOR MICROSOFT C, SMALL C, AND LATTICE C COMPILERS

Small-Windows: A Library of Windowing Functions for the C Language



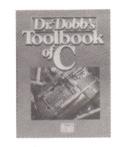
mall-Windows is a complete windowing library for C. The package contains: 18 video functions written in assembly language; 7 menu functions that support both static and pop-up menus; and 41 window functions, including functions to clean, frame, move, hide, show, scroll, push, and pop windows. A file directory facility illustrates the use of the window menu functions and provides file selection, renaming, and deletion capability. Two test programs are also included. For PC/MS-DOS systems, Microsoft C Version 4.0, Small-C, and Lattice C compilers. Documentation and full source code is included.

Small-Windows

Item #35-6

\$29.95

Dr. Dobb's Toolbook of C



This authoritative reference contains over 700 pages, including the best C article from Dr. Dobb's Journal along with new material. You'll find hundreds of pages of valuable C source code, including a complete compiler, an assembler, and text processing programs.

Toolbook of C

Item #005 \$29.95

ORDER FORM

TO ORDER:

Return this order form with your payment to: M&T Books 501 Galveston Dr. Redwood City, CA 94063 Or, call TOLL-FREE 800-533-4372 Mon-Fri 8AM-5PM In CA call 800-356-2002

Name		
Address		
City	_Stαte	Zip
Item # Description		Price
	Subtotal	
CA residents add so	ales tax %	
Add \$2.25 per iten	n for shipping	

TOTAI.

For Disk Orders, please indicate format. Refer to product description for standard format availability.

_ MS-DOS CP/M _ Kaypro _ 8" _ Osborne _ Apple _ Zenith Z-100	
For Small-Windows, indicate: Microsoft version 4.0 compiler Small-C compiler	
Lattice C compiler	

Check	enclosed.	Make	payable	to	M&T
Publishing.					

Charge my VISA	M/C	Amer. Exp.
Card No	Ехр.	
Signature		

CODE: 3131A

On Command: Writing a **Unix-Like Shell** for MS-DOS

by Allen Holub

his book and ready-to-use program demonstrate how to write a Unix-like shell for MS-DOS, with techniques applicable to most other programming environments as well. The book and disk include a detailed description and working version of the shell, complete C source code, a thorough discussion of low-level DOS interfacing and significant examples of C programming at the system level. Supported features: read, aliases, history and C-Shell-based shell scripts. The Unix-like control flow includes: if/then/else; while; foreach; switch/case; break; continue. For IBM PC and direct compatible's. All source code included on disk

On Command

Item #29-1 \$39.95

hen used with the **shell**, this collection of utility programs and subroutines provide you with a fully functional subset of the Unix environment. Utilities include: cat; cp; date; du; echo; grep; ls; mkdir; mv; p; pause; printevn; rm; rmdir; sub; and chmod. Complete source code and manual included. Item #12-7 \$29.95

Program Interfacing

by William Wong

riginally featured in Micro/Systems Journal, Program Interfacing to MS-DOS provides ten concise articles that will orient any experienced programmer to the MS-DOS environment. All source code discussed is also contained on disk.

Topics include: program construction, character base input and output functions, and file access. You'll also find a discussion of CP/M style vs. Unix-style DOS file access, sample program files, and a detailed description of how to build device drivers. A device driver for a memory disk and a character device driver are provided on disk with full source code.

Interfacing

to MS-DOS

Item #34-8 \$29.95

is a text formatter that is written in C and is compatible with the Unix NROFF. It includes complete implementation of the -ms macro package, and an in-depth description of how -ms works. NR does hyphenation and simple proportional spacing, and supports automatic table of contents generation and indexing, automatic footnotes and endnotes, italics, boldface, overstricking, understricking, and left and right margin adjustment. Also: extensive macro and string capability, number registers in various formats, diversions and diversion traps, input and output line traps. Full source code included. For PC compatibles.

Item #33-X \$29.95

how to customize CONFIG.SYS and use ANSI.SYS to change the appearance of DOS. You'll find extensive batch file coverage with example routines that use redirection operators, filters and pipes, and ready-to-use assembly language programs that enhance DOS. Full source code is included.

Taming MS-DOS

aming MS-DOS takes you beyond the

basics, picking up where your DOS

manual leaves off. You'll learn how to create

a memory-resident clock, how to rename

subdirectories and change file attributes,

how to create AUTOEXEC.BAT files, and

by Thom Hogan

Taming MS-DOS

Item #24-0 \$19.95

Taming MS-DOS with disk

Item #59-3 \$34.95



BUSINESS REPLY MAIL

FIRST CLASS PERMIT 871 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

M&T Books

501 Galveston Dr. Redwood City, CA 94063

NO POSTAGE **NECESSARY** IF MAILED IN THE UNITED STATES



XOR CHAIN

Listing One (Listing continued, text begins on page 36.)

```
j->llink ^= i ^ s->child;
else
                                      /* Case 4 */
    j = i->llink ^ s->child;
    if (j-\rangle rlink == i)
                                     /* Case 4a */
         * adjust the pointers for s->child,
         * i's parent
        if (i->key < s->child->key)
            s->child->llink = j ^ s->parent;
             s->child->rlink = j ^ s->parent;
         * adjust the pointers for i's children
        j->llink ^= i ^ s->child;
        j->rlink = i->rlink;
        k = i->rlink ^ s->child;
k->llink ^= i ^ j;
        k->rlink ^= i ^ j;
                                     /* Case 4b */
    else
```

(continued on next page)





the dBx Translator

- C from dBASE II, III, III+ programs
- Move to UNIX, XENIX, QNX, MAC, AMIGA
- · Faster, more reliable IBM PC programs
- Supports multi-user and network
- Run your code on any standard C system
- · Know dBASE? Learn C easily.
- Priced from \$350; available from distributors
- Includes full screen handler library
- · Supports a choice of C database managers



1720 Post Road East, Westport, CT 06880 Telephone: 203-255-3400 • Telex: 6502972226MCI MCIMAIL-DESKTOPAI

dBASE is a trademark of Ashton-Tate

dBx is a trademark of Desktop Al

CIRCLE 258 ON READER SERVICE CARD

8031

FORTH DEVELOPMENT ENVIRONMENT

Take advantage of Bryte's tools to make your job easier:

- Bryte's development environment uses BRYTE-FORTH on the actual production hardware during product development. No emulators, no changes, no surprises.
- Optional PC-based cross-development tools use DOS files as microcontroller mass storage. These files can be used to generate compact EPROM images, detailed listings, and cross-references.

Why not start developing the Bryte way today?

BRYTE-FORTH 8Ø31 EPROM 100.00 (includes 130 page User's Manual) Utility disk(s) 65.00* Cross-compiler/Cross-assembler 235.00* 1000.00* 8031 unlimitted quan. license

* Includes complete source code

bryte computers, inc.

P.O. Box 46 Augusta, ME 04330-0046

207/547-3218

CIRCLE 387 ON READER SERVICE CARD

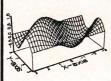
Publication Quality Scientific Graphics

Over 125 C routines make scientific plotting easy

- → linear, log, & polar plots
- → bar charts & Smith charts
- → contour plots with labels
- → 3-D curves, 3-D surfaces
- → 4 curve types, 8 markers, errorbars
- → 14 fonts, font editor
- → unlimited levels of superscripts
- → 4096 x 3120 resolution in 16 colors on EGA, Tecmar, Sigma boards
- → zoom, pan, window and merge plots
- → high resolution printer dumps

SOURCE INCLUDED for personal use only \$350. Demo \$8

256k, IBM, AT&T, Corona PCs, DOS 2.xx, 3.xx Most boards, printers, and plotters supported Microsoft, Lattice, DeSmet, Aztec, C86 compilers







Scientific Endeavors Corporation

Route 4, Box 79 Kingston, TN 37763 (615) 376-4146

CIRCLE 210 ON READER SERVICE CARD

M Street Software

80386 Support!

SCRUTINY

Advanced symbolic debugger.

- Multi-language: compatible with Turbo Pascal, Microsoft Assembler, others.
- Multi-DOS: works with all MS-DOS/PC-DOS computers.
- Multi-level: debug at source level and machine level, separately or together.
- Multi-display: debug character-mode and graphicsmode programs, with movable debug windows.
- Multi-chip: support for 8086, 80186, 80286, 80386.
- Fast 80386 "memory breakpoints" (stop program when specified variable is accessed or modified).

Scrutiny/Master \$99.95

for debugging Turbo Pascal, Microsoft Assembler, and other languages.

Scrutiny/Turbo Special price! \$49.95 for debugging Turbo Pascal only.

VISA/MC AMEX accepted. In Texas please add sales tax. Outside of North America add \$10 per item shipping.

M Street Software 5400 E. Mockingbird Lane Suite 114 Dallas, Texas 75206 214-827-4908

Information also available via our 24 hour 300/1200 modem: 214-669-1882.

CIRCLE 275 ON READER SERVICE CARD

XOR CHAIN

Listing One

```
(Listing continued, text begins on page 36.)
 * locate the replacement item
t = &temp;
Associate(t,s->root);
t->parent = s->child;
t->child = i;
GoLeft (t);
while ( t->child->rlink != t->parent )
    GoRight (t);
  adjust the pointers to free t->child
t->parent->rlink ^= t->child->llink ^
                       t->parent
                       t->child;
if (t->child->llink != t->parent)
    k = t->child->llink ^ t->parent;
    k->llink ^= t->parent ^ t->child;
    k->rlink ^= t->parent ^ t->child;
 * adjust the pointers for s->child,
 * i's parent
if (i->key < s->child->key)
    s->child->llink = t->child ^ s->parent;
else
    s->child->rlink = t->child ^ s->parent;
t->child->llink = i->llink;
t->child->rlink = i->rlink;
             * adjust the pointers for i's children
             j->llink ^= i ^ t->child;
             i->rlink ^= i ^ t->child;
             k = i \rightarrow rlink ^ s \rightarrow child;
            k->llink ^= i ^ j;
             k->rlink ^= i ^ j;
    DropItem(i);
```

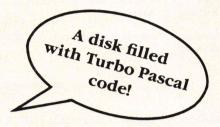
End Listing

Turbo Tech Report Speaks Your Language.









The newsletter/disk publication for Turbo Pascal® users

Are you looking for powerful utilities written in Turbo Pascal that you can use to develop software or incorporate into your programs? Are you interested in improving and expanding your Turbo Pascal programming skills?

Then you deserve a subscription to *Turbo Tech Report*, the bimonthly newsletter/disk publication from the publishers of *Dr. Dobb's Journal* and *Micro/Systems Journal*. Each issue delivers more than 250K of Turbo Pascal source code programs on disk, and 24 + pages of articles, Turbo Pascal software and book reviews, and

analysis and commentary. It's the only publication delivering such focused technical articles with code on disk. Each valuable issue contains:

- Articles on topics like speedy 3D graphics, mathematical expression parsers, creating global gotos, memory resident and AI applications and more—all written by Turbo experts.
- Reviews of the latest Turbo Pascal software programs from companies like Borland International, Blaise

Computing, Media Cybernetics, Nostradamus, and more!

- *News and commentary* detailing the latest products and developments in the Turbo Pascal programming community.
- A disk filled with Turbo Pascal code! You'll get the Turbo Pascal utilities and routines discussed in the newsletter's articles, as well as applications developed by Turbo users from around the world. You'll receive programs that make labels, generate menus, provide faster screen access, transfer files, and more!

If you're an expert Turbo Pascal programmer or a novice interested in expanding your Turbo skills, you need a publication that speaks your language: Turbo Tech Report. Subscribe today at the special price of just \$99—that's 33% off the regular price of \$150. To order by credit card, call tollfree 1-800-533-4372. Or mail the coupon with your payment to Turbo Tech Report, 501 Galveston Drive, Redwood City, CA 94063.

issues with 6 disks) for \$99. Format: PC/MS-DOS Macintosh CP/M: Kaypro Osborne Apple PAYMENT MUST ACCOMPANY ALL ORDERS Check/money order enclosed. Charge my: VISA M/C Amexp. Card # Exp. Signature Name Address	— Yes! I want a one-year subs	cription to Turbo Tech Repo	ort (
CP/M:	issues with 6 disks) for \$99.		
PAYMENT MUST ACCOMPANY ALL ORDERS Check/money order enclosed. Charge my: VISA M/C AmExp. Card # Exp Signature Name	Format: PC/MS-DOS	Macintosh	
☐ Check/money order enclosed. ☐ Charge my: VISA M/C AmExp. Card # Exp Signature Name Exp.	CP/M: Kaypro O	sborne Apple	
☐ Check/money order enclosed. ☐ Charge my: VISA M/C AmExp. Card # Exp Signature Name			
☐ Charge my: VISA M/C AmExp. Card # Exp Signature Name	PAYMENT MUST ACCOMPA	NY ALL ORDERS	
Card # Exp Signature Name	Check/money order enclo	sed.	
SignatureName	Charge my: VISA _	M/C AmExp.	
SignatureName			
Name			
	Signature		
Address	Name		
	Address		
City State Zip	City	_ State Zip	

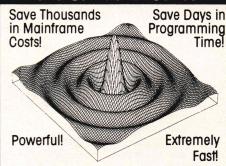
CIRCLE 119 ON READER SERVICE CARD

DEVICE DRIVERS

```
Listing One (Text begins on page 44.)
; prndrv.asm
         printer driver startup code
         ms/pc-dos 2.x, 3.x installable device driver
;copyright (c) Andy Klein 1987
PRIVATE_STACK_SIZ equ 256 ;private stack size, probably much larger
                               ; than necessary
IS_CHAR DEV equ 32768
                              ;bit to set for this driver
codeseg
           segment para public 'CODE'
codeseg
dataseg
           segment para public 'DATA' ; define first
dataseg
           ends
              cs:codeseg, ds:dataseg, es:dataseg, ss:dataseg
           segment para public 'CODE'
codeseg
    org
                    ;drivers are always org'ed at 0
                     ; with the device header first
public prn_driver, dev_strategy , dev interrupt
prn_driver
                     far ;drivers invoked as far calls by DOS
              proc
        ; device header starts here
next dev db 4 dup (255)
                                 ;no next driver in this file, need a
                                 ; (long) -1 which is 4 bytes of 255
attribute dw
                 IS CHAR DEV
strategy dw
                 dev_strategy_ ;device strategy entry point
interrupt dw
                 dev_interrupt_ ;device interrupt entry point
dev name db
                  'PRN
        ; end of the device header
    ; code segment variables, these will be addressable before we setup
    ; our data and stack segment
                 (0) ; pointer to request header, segment part(0) ; pointer to request header, offset part
req hdr seg dw
req hdr off dw
                        ; caller's ss
caller ss dw
                    (0)
caller_sp dw
                    (0)
                         ; caller's sp
        ;strategy - this strategy function saves a long pointer to a
        ; request header in code segment variables reg hdr seg and
        ; req hdr off.
        ; The address of the request header is passed in es:bx
dev_strategy_:
    mov word ptr cs:req hdr off,bx
                                       ; save request header pointer
    mov word ptr cs:req hdr seg,es
                                       ; in code segment variables
                                        ; ...and back to dos
public $cswt, $begin
                        ; keep Aztec linker happy, all Aztec C compiled
                         ; functions have a reference to these 2 functions
                         ; which normally drag in the startup code
    $cswt proc near
    $cswt endp
    $begin proc near
    $begin endp
```

```
extrn
             driver functions :near
    ; interrupt - not a true interrupt handler (it ends with a ret as
    ; to an iret), this function recieves no parameters. Instead it
    ; uses the information stored in the request header we recieved
    ; saved a pointer to in the strategy function to determine what
                                                                to do.
dev interrupt :
            ;STEP 0
            ; Preserve machine state
    cli
                   ; no interrupts when dealing with machine state
   push ds
                    ; save machine state on caller's stack
   push es
   push ax
   push bx
   push cx
   push dx
   push si
   push di
   push bp
   pushf
   mov cs:caller ss,ss
                            ; save caller stack frame (ss and sp)
    mov cs:caller_sp,sp
                            ;interrupts OK
            ; Now the real work starts ...
            ; Get the driver's segment into the ax register
            ; This begins our task of establishing addressability
    mov ax, cs
           ;STEP 2
            ; Copy the request header stored at req hdr seg:req hdr
            ; into our private and C addressable request header
   mov si, cs: req hdr off
                                 ; load up the source string segment:
                                                                  offset
   mov bx, cs: req hdr seg
                                 ; into ds:si registers
   mov ds, bx
   mov es.ax
                                 ; load up the destination string
                                                         segment:offset
   mov di, offset cs:driver rh ; into es:di registers
                                 ;set direction flag to increment
   cld
                                                               si and di
   xor cx.cx
                                ;clear out cx
                                ; first byte of request header is its
   mov cl, [si]
                                                                 length
                                ; ... and copy it ...
   rep movsb
           ;STEP 3
           ; This step establishes addressability of data segment
              and sets up driver's stack frame.
             It also sets register bp equal to sp, a state required
              for calling the first C function
                   Remember that register ax contains the driver
                                                                 seament
                   We will set ds, es, ss all equal to cs ... 8080
   mov bx, offset c stack top
                                ;this will go into sp
                     ; no interrupts while we mess with these registers
   cli
   mov ds, ax
   mov es.ax
   mov ss, ax
                     ; now establish our stack frame
   mov sp, bx
                     ;bp = sp this is critical for C
   mov bp, bx
                                                 (continued on next page)
```

for IBM PC-XT-AT-System/2 and Compatibles written by Lee E. Edlefsen and Samuel D. Jones



Easy to Learn and Easy to Use!

The New Standard for Scientific and Statistical Computation

"I used to use FORTRAN and PASCAL for languages, TSP and Minitab for statistics, MATLAB for math, and NAG and IMSL for FORTRAN subroutines. Now I just use GAUSS.

> Dr. Choon-Geol Moon Stanford University

- STATISTICS (means, frequencies, crosstabs, regression, nonparametrics, general max likelihood, non-linear least squares, simultaneous equations, logit, probit, loglinear models, & more)
- GRAPHICS (publication quality 2D & 3D: color, hidden line removal, zoom, pan; up to 4096 x 3120 resolution; produce Tektronix format files; output to most screen drivers, plotters, printers)
- · PLUS:
- . DATABASE MANAGEMENT
- SIMULATION
 TIME SERIES/SIGNAL PROCESSING
 - · LINEAR PROGRAMMING
 - NON-LINEAR OPTIMIZATION
 - NON-LINEAR EQUATION SOLUTION
 - INTERACTIVE MATRIX PROGRAMMING LARGE-SCALE MODULAR PROGRAMMING
 - ADD YOUR OWN COMMANDS
 - · LINK FORTRAN, C, ASSEMBLER SUBROUTINES

Buy the GAUSS Programming Language by itself or as part of the GAUSS Mathematical and Statistical System, which includes 2D & 3D graphics plus over 200 applications programs written in the GAUSS Programming Language for doing a variety of mathematical, statistical, and scientific tasks. Full source code is provided with these programs.

APTECH SYSTEMS, INC. (206) 631-6679

P.O. Box 6487 Kent, WA 98064

30 DAY MONEY-BACK GUARANTEE

The GAUSS Programming Language (alone)
--

DOS 2.10+, and a math coprocessor.

NOT COPY PROTECTED

DEVICE DRIVERS

```
Listing One (Listing continued, text begins on page 44.)
                       ; ok for an interrupt to occur
             ; now we have set up the environment for C
             :STEP 4
             ; Call our first C function,
             ; passing the command code from the request header as
             ; a parameter. (void) driver_functions (cmd_code);
     xor ax.ax
                                 ; clear out ax
     mov bx, offset driver rh
                                 ;bx points to our request header
     mov al, byte ptr [bx + 2]
                                 ;3rd byte of request header is command code
     push ax
                                 ; to pass parameter(s) to a C function,
     call driver functions
                                 ; push it (them) on stack and call function
     pop ax
                                 ; caller must remove parameter(s) from stack
             ;STEP 5
             ; At this point we are done with the function our driver
             ; was to perform. Now we must copy our private request header
             ; back into the original request header DOS gave us a pointer
             ; to back in the strategy function.
     mov es, cs: req hdr seg
                                  ;load address for orig RH into es:di
     mov di, cs: req hdr off
     mov si, offset driver rh
                                  ;ds:si our (updated) copy
     cld
                                  ; set direction to increment
     xor cx, cx
                                  ;clear out cx
     mov cl, [si]
                                  ; length is at first byte
    rep movsb
                                  ; ... and copy it
             ;STEP 6
             ; Restore machine state saved in STEP 0
             ; and return to DOS
                            ; machine state restore should not be interrupted
    mov ax,cs:caller ss
                            ; switch to caller's stack frame
     mov ss,ax
     mov ax, cs:caller sp
    mov sp, ax
     popf
                           ; ... pop all of it ...
    pop bp
     pop di
     pop si
     pop dx
     pop cx
    pop bx
     pop ax
     pop es
     pop ds
     sti
                            ; enable interrrupts
     ret
                            ; ... back to dos and bye bye
 prn driver
             endp
                           ;end of driver code
 codeseg
            ends
     ;Data segment, contains our local stack space
     ; and C addressable copy of request header
 dataseg segment para public 'DATA'
     public c stack top
     db PRIVATE STACK SIZ dup (0)
 c stack top label word
     public driver_rh_
 driver_rh_ db 32 dup (0)
                             ;C addressable copy of request header
dataseg
           ends
END
                                                                                           End Listing One
```

Listing Two

```
/** rh.h
    Device Driver Request Header structure definition
    and status word #define's
copyright (c) 1987 Andy Klein
typedef struct
    unsigned char length,
                                /* length of header */
                                /* unit code ... which unit to use */
                  unit,
                  cmd:
                                 /* command to execute */
                                /* status of operation */
    unsigned int status;
    unsigned char reserved[8],
                  media_type;
                                /* media descriptor byte, block dev only */
    unsigned int
                  xfer buf offset,
                  xfer buf segment,
                  xfer count;
                  dummy [32 - 20];
    char
                                 /* data for operation */
} request hdr;
#define
         ERROR MASK
                            32768
        BUSY MASK
                            1024
#define
#define
         DONE MASK
                            512
#define WRITE PROTECTED
                            0x00
#define UNKNOWN UNIT
                            0x01
#define DEV NOT READY
                            0x02
#define UNKNOWN CMD
                            0x03
#define
        CRC ERROR
                            0x04
        BAD DRIVE REQ LEN
#define
                            0x05
#define SEEK ERROR
                            0x06
```

(continued on next page)

386 DEBUG

- A symbolic debugger for 80386 32-bit protected mode programs which run under Phar Lap's 386 DOS-Extender™
- Breakpoints, data watchpoints, and built-in disassembler
- Fully compatible with Phar Lap's 386 ASM/LINK, the MetaWare 80386 High C™ and Professional Pascal™compilers, and the Green Hills 80386
 Fortran compiler
- Runs on all DOS-based PCs equipped with an 80386 CPU, including the Compaq® DESKPRO 386™, the IBM®PS/2™ Model 80, and most accelerator cards, including the Intel Inboard™ 386/AT
- \$195-Available today

(617) 661-1510

Phar Lap Software, Inc. 60 Aberdeen Ave. Cambridge, MA 02138



"The 80386 Software Experts"

CIRCLE 343 ON READER SERVICE CARD



YOUR WAY CLEAR INTO THE FUTURE WITH THE VIRTUAL G.DOS/GRAM
ENVIRONMENT

UNIX USERS SAY UNIX

IS A STEP DOWN FROM BOOS! UNBELIEVABLE ?

THE COL DESKTOP MINIFRAME 22 BIT VIRTUAL MEMORY IN RAM

A SUBCONSCIOUS BODY OF MULTITASKING TRAFS & VECTORS THAT RUNS UNLIMITED TASKS; FULL SCREEN COMPRESSION CACHING WITH CONTROL-C; CONCURRENT TURBO SUPERBASIC INTERPRETING THAT'S MORE STRUCTURED THAN C; UNLIMITED LENGTH STRINGS, BUFFERS & PROGRAM LINES WITH NO LOSS OF VARS WHEN EDITING SOURCE CODE; CONSOLE WINDOWING...

THE PERFECT TOOL FOR DEVELOPING AND IMPLEMENTING C LANGUAGE CONSTRUCTS!

CALL 201-328-8846

OR WRITE FOR THE LATEST CATALOG-DIRECTORY

LAMILUN LUNGUING DOVER, N.J. 02801

CIRCLE 144 ON READER SERVICE CARD

DEVICE DRIVERS

```
Listing Two (Listing continued, text begins on page 44.)
```

```
#define UNKOWN_MEDIA 0x07
#define SECTOR_NOT_FOUND 0x08
#define PRN_NO PAPER 0x09
#define WRITE_FAULT 0x0A
#define READ_FAULT 0x0B
#define GENERAL_FAILURE 0x0C
#define INVALID_DISK_CHG 0x0F
```

End Listing Two

Listing Three

```
/** drvfunc.c
        Device Driver for DOS 2.x, 3.x
        This is the function table that is called once
        the interrupt function has established addressability
 copyright (c) 1987 Andy Klein
 #define LAST FUNCTION 15
 #define Q_FUNCTIONS LAST_FUNCTION + 1
extern void init(), output_status(), output(), output_flush(),
    output_verf(), bad_cmd();
     /**
        function_table is an array of pointers to functions
        returning nothing (void). It is analogous to a jump table
        in assembler.
    **/
void (* function_table[Q_FUNCTIONS])() = {
    /* 0 */
                  init,
    /* 1 */
                bad cmd,
                                    /* media check */
    /* 2 */
                bad cmd,
                                    /* build bpb */
    /* 3 */
                bad cmd,
                                   /* ioctl input */
                                   /* input */
    /* 4 */
                bad_cmd,
    /* 5 */
                           /* input no wait */
/* input status */
                bad cmd,
    /* 6 */
                bad cmd,
    /* 7 */
                                   /* input_flush */
                bad_cmd,
    /* 8 */
                output,
    /* 9 */
                output verf,
    /* 10 */
                output status,
    /* 11 */
                output_flush,
    /* 12 */
                bad cmd,
                                    /* ioctl output */
    /* 13 */
                bad cmd,
                                    /* open device */
    /* 14 */
                bad cmd,
                                   /* close device */
    /* 15 */
                bad cmd
                                    /* removable_media */
1:
void
driver functions (cmd)
int cmd:
    if ( cmd > LAST FUNCTION )
        (void) bad_cmd();
        (void) (* function_table[cmd])();
}/* driver_functions() */
Listing Four
```

End Listing Three

```
/** error.c
    Error handler for printer driver
copyright (c) 1987 Andy Klein
**/
#include "rh.h"
extern request_hdr driver_rh;

void
bad_cmd()
{
    driver_rh.status = ERROR_MASK | UNKNOWN CMD;
```

(continued on page 74)

Hard Locks for Soft Parts.



At Rainbow Technologies, we think protecting software developers' investments is very serious business. That's why we designed the first fully effective security solution for software running on PCs and other computers.

Our family of virtually impenetrable Software Sentinel hardware keys provides the highest level of software protection the developer can get. While remaining invisible to the end user.

Take a look.

Key Sentinel Family Features.

Prohibits unauthorized use of software \square No need for copy protection \square Unlimited backup copies \square Virtually unbreakable \square Pocketsize key \square Transparent operation \square Transportable

Software Sentinel.

- Runs under DOS and Xenix, on IBM PC/XT/AT and compatibles
- O Algorithm technique (Never a fixed response)
- O Serial or parallel port version
- O Minimal implementation effort
- O Higher level language interfaces included
- 100 times faster than fixedresponse devices (1ms)

Software Sentinel-C.

- O For developers who want to customize or protect multiple packages with one device
- o 126 bytes of non-volatile memory that is programmed before shipment of software
- We supply a unique programming adapter for programming the unit

- O Higher level language interfaces included
- O Runs under DOS on PC/XT/AT and compatibles
- O Parallel port version only

Software Sentinel-W.

- O Designed for workstations, supermicros and minicomputers
- O Serial port only (modem-type)
- O Algorithm technique
- We provide detailed interface specifications: Developer creates a port driver
- O Interface requirements: 25 pin DB25P or DB25S; RS232/RS422/RS423
- Only signals used: DTR & RTS from computer; signal ground; DSR or optional DCD from Software Sentinel-W or external device. TXD, RXD, CTS, RI passed through.

Call For Software Sentinel Evaluation Kit Pricing.



CIRCLE 255 ON READER SERVICE CARD

The Heap Expander

- dynamically allocates data storage space in expanded memory
- simple interface

 up to 8 megabytes of heap space \$59.95

- with appropriate hardware libraries and source code for:
 - -Microsoft C, Lattice C, Turbo C,
 - Mark Williams C, and others
 - Turbo Pascal
 - Logitech Modula-2
- requires IBM PC, XT, AT, or close compatible with LIM-standard expanded memory and MS—DOS or PC—DOS ver. 2.0 or above
- MC/VISA/COD call
 - 1-800-248-1045 x100 (US)
 - 1-800-952-5560 x100 (Idaho)

The Tool Makers

P.O. Box 8976 Moscow, Idaho 83843 (208) 883-4979

*Idaho residents add 5% sales tax

CIRCLE 319 ON READER SERVICE CARD



FORTRAN/RATFOR TO C TRANSLATOR*

- Maximize the vast resources of FORTRAN while moving up to C. Speed up new C development and avoid resignating the wheel
- inventing the wheel.

 Use RTC Plus to translate FORTRAN code and libraries and maintain code with greater ease and flexibility in C.
- Source code to C libraries is included.
- RTC Plus supports standard FORTRAN-77 as well as some DEC VAX extensions (excluding FORTRAN I/O, character and complex statements/expressions). Over 95% of STUG's RATFOR is supported. The Translator generates K&R C.
- Finally a cost-effective method of conversion into C.



DEMO \$10 MS-DOS \$450

*Translate: "To convey to heaven without natural death."

COBALT BLUE

1683 MILROY, SUITE 101, SAN JOSE, CA 95124 408-723-0474

CIRCLE 370 ON READER SERVICE CARD

DEVICE DRIVERS

```
Listing Four (Listing continued, text begins on page 44.)
#define TIME OUT 1
#define IO ERR
                  8
#define NO PAPER 32
#define BUSY
                  128
void
driver error (stat)
int stat;
    int err code;
    if ( stat & ERROR_MASK )
        stat ^= ERROR MASK;
    switch ( stat )
        case TIME_OUT:
                           err_code = DEV_NOT_READY;
                           break;
        case IO ERR:
                           err_code = GENERAL_FAILURE;
                           break;
        case NO PAPER:
                           err_code = PRN_NO_PAPER;
                           break;
        case BUSY:
                           err code = DEV NOT READY;
                           break;
        default:
                           err_code = GENERAL_FAILURE;
                           break;
    driver_rh.status = ERROR_MASK | err_code;
}/* driver_error() */
                                                                                            End Listing Four
Listing Five
/** init.c
        Printer driver initialization
copyright (c) 1987 Andy Klein
#include "rh.h"
extern request_hdr driver_rh;
char *title = "\nPrinter Device Driver for Okidata 92";
char *copyw = "copyright 1987 (c) Andy Klein\n";
void
init()
    extern unsigned Uend;
              /* _Uend is inserted by the Aztec linker */
    unsigned show cs();
   void reset_printer(), initialize_oki92();
    puts(title);
    puts (copyw);
    (void) reset_printer();
    (void) initialize oki92();
    /* set ending address of driver, set status word */
driver_rh.xfer_buf_segment = show_cs();
    driver_rh.xfer_buf_offset = (unsigned int) & Uend;
    driver_rh.status = DONE MASK;
} /* init() */
#define ELITE FONT 28
void
initialize_oki92()
    char_2_prn(ELITE_FONT);
}/* initialize_oki92() */
                                                                                            End Listing Five
```

74

Listing Six

```
; show_cs.asm
; c call is (unsigned) show_cs();
; Returns contents of cs register
; copyright (c) 1987 Andy Klein
include lmacros.h

procdef show_cs
mov ax, cs
pret
pend show_cs
finish
```

End Listing Six

Listing Seven

```
Printer driver output functions
copyright (c) 1987 Andy Klein
#include "rh.h"
extern request hdr driver rh;
output ()
    int stat;
    unsigned xfer off, bytes xferd;
    register char outch;
    char fetch char();
    xfer off = driver rh.xfer buf offset;
    for ( bytes_xferd = 0; bytes_xferd < driver_rh.xfer_count;
            ++bytes xferd, ++xfer_off )
        outch = fetch_char(driver_rh.xfer_buf_segment, xfer_off);
        if ( (stat = char_2 prn(outch)) )
             (void) driver error(stat);
            break:
        }
    }
    driver_rh.xfer_count = bytes_xferd;
    if (! stat)
        driver_rh.status = DONE MASK;
        driver rh.status = stat;
}/* output() */
void
output verf()
     (void) output();
 }/* out_verf() */
 void
 output status()
         if device is currently doing an operation,
         driver rh.status = BUSY_MASK & DONE_MASK;
         else if a write can begin immediately,
         driver_rh.status = 0 & DONE MASK;
     if ( printer busy() )
         driver_rh.status = BUSY_MASK & DONE MASK;
     else
                                           (continued on next page)
```

Amouncing

Version 2.2 of the



The power and flexibility of UNIX commands within the DOS environment.

4 Reasons

Why the MKS Toolkit Is a Very Large Package for a Small Price:

1. It contains the UNIX full-screen editor VI/EX

 and handles the various national character sets provided with DOS, as well as 8-bit data and improved support for EGA and colour attributes.

2. It comes with a complete KORN SHELL

— a programming language in itself including **vi** and **emacs** command-line editing mode.

3. It has the only version of AWK available under DOS

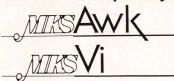
— written to the latest Bell Labs specifications for System V.3, allowing multiple-subscripted arrays; **awk** is an excellent fourth generation language that even non-programmers will find readily accessible.

Besides all this it comes with over 110 programs

— including many new commands such as **init**, **login**, **passwd**, and **who** to facilitate multiple users of the same machine, or multiple application environments; **pr** and **fmt** for formatting files; **crypt** for file encryption; **pack**, **unpack**, and **pcat** for data compression; and much more.

All for \$139.

Now available separately:



\$75 each

The MKS Toolkit, site-licensed to major American corporations, is designed for IBM PCs, XTs, ATs, and compatibles running under MS-DOS (or PC-DOS) 2.0 and later. It includes over 380 pages of documentation.

Mortice Kern Systems Inc.

43 Bridgeport Road East, Waterloo, Ontario, Canada N2J 2J4 (519) 884-2251 uucp: {allegra,decvax,ihnp4}!watmath!mks!toolkit

For information or ordering call collect.

Prices quoted in U.S. funds. MasterCard, VISA, American Express, and purchase orders accepted. OEM & dealer inquiries invited. UNIX is a trademark of Bell Labs. MS-DOS is a trademark of Microsoft Corp. No UNIX licence required. Updates to existing licences are available for \$45.00.

DEVICE DRIVERS

```
Listing Seven (Listing continued, text begins on page 44.)
        driver rh.status = DONE_MASK;
}/* output status() */
void
output flush()
        terminate all pending requests,
        meaningless for this device
    driver_rh.status = DONE_MASK;
}/* output_flush() */
                                                                                        End Listing Seven
Listing Eight
   prlport.asm
     low level parallel port output functions
;copyright (c) Andy Klein 1987
include lmacros.h
                        ; Aztec C macros for assembly language functions
                        ; for other environments replace with calling
                        ; conventions for your complier
ERROR MASK equ
                    32768
PRINTER_INT equ
                    017H
PRINTER ID equ
                    0
                                 ; lpt1 = 0, lpt2 = 1, lpt3 = 2
    ;printer commands, put into ah
```

ould you like copy protection and customer satisfaction?



here's a better way to protect your software. It's called the Secom Key, and it works.

- The Key is completely transparent to the end user.
- ☐ Won't interfere with peripheral operations.
- Doesn't occupy the disk drive.
- ☐ The Key allows <u>unlimited backup copies.</u>
- ☐ Makes site licensing easy and auditable.
- ☐ Easily installed. Uses only 1000 bytes.
- Over 60,000 have been sold worldwide.
- ☐ Same size as RS-232 plug.
- ☐ Available in quantities for as low as \$19.95.



or more information, contact Secom Information Products Co.



500 Franklin Square 1829 East Franklin Street Chapel Hill, NC 27707

The Secom Key.. for <u>real</u> software protection.



Secom Information Products Company
A Subsidiary of Secom General Corporation

Call Toll-free 1-800-843-0413

CIRCLE 394 ON READER SERVICE CARD

```
PRINT CHAR
                    0
             equ
INIT PRN
                    1
             equ
READ STAT
             equ
    ;printer status byte error codes, returned in ah
TIME OUT
            eau
                   1
IO ERR
            equ
                   8
NO PAPER
                   32
            equ
                   128
BUSY
            equ
PRINTER FAILURE
                  equ
                        TIME OUT+IO ERR+NO PAPER
    ;C call is (void) reset_printer();
    ;no value returned
procdef reset_printer
        mov ah, INIT PRN
        mov dx, PRINTER ID
        int PRINTER INT
        xor ax, ax
        pret
pend reset_printer
    ;C call is printer_busy();
    ; returns 0 (FALSE) if not busy, nonzero (TRUE) if busy
procdef printer busy
        mov ah, READ STAT
        mov dx, PRINTER ID
        sti
        int PRINTER INT
        and ah, BUSY
        jz printer_is_busy
```

(continued on next page)

AN ADVANCED APL TEXT IPI IPI Are you getting the most productivity possible out of the most productive programming language? If not, this book is for you. APL ADVANCED TECHNIQUES AND UTILITIES Disk \$15.00 450 pp. \$44.95 Branching and Looping Writing Reports • Computer Efficiency Considerations • Programming Standards Positioning Character Data **Workspace Design and** Sorting and Searching **Documentation** File Design and Utilities Selecting Frequency Counts, Accumulations and Cross Tabulations • Boolean Techniques Irregular Arrays Writing User-Friendly Interactive **Curve Fitting** • Financial Utilities Date Manipulations • Exception Handling • System Development Procedure More than 150 time-tested utility functions listed in the book and available on floppy disk. TO ORDER, call (203) 872-7806 ZARK INCORPORATED 53 SHENIPSIT STREET VERNON, CT 06066

Beyond the 64K Boundary With Automatic Overlays

64180/Z80

Using bank switched or disk based overlays, your modular program can be as large as you want. Our linker generates the code to switch in the correct bank at the right time, or load the right module from disk. Includes source to the overlay loader for easy customization to your requirements. SLRNK Plus 3.0, from the leaders in 8-bit development tools.

\$195

_SLR_Systems

Butler, PA 16001 (412) 282-0864 (800) 833-3061 TELEX: 559215

CIRCLE 78 ON READER SERVICE CARD

DEVICE DRIVERS

```
Listing Eight (Listing continued, text begins on page 44.)
```

```
pret
    printer is busy:
        mov ax, 1
        pret
pend printer busy
       C call is
                    (char) fetch char(segm, offs);
         gets the character at segm:offs and
         returns it
procdef fetch char, << segm, word>, <offs, word>>
        push ds
        mov bx, offs
        mov ds, segm
        mov al, byte ptr ds:[bx]
        and ax, OffH
        pop ds
        pret
pend fetch_char
    ;C call is char_2_prn(outch);
         prints outch on PRINTER ID
    ; returns 0 if ok, error code
procdef char 2 prn, <<outch, byte>>
        mov al, outch
```

mov ah, PRINT_CHAR mov dx, PRINTER ID

386
C and Pascal

MetaWare Incorporated announces the first available C and Pascal compilers that generate

protected-mode 80386 code

for running on any 80386 machine that runs MS-DOS (e.g., the Compaq Deskpro 386). The compilers are functionally identical to the well-respected 8086/286 MS-DOS High C™ and Professional Pascal™ compilers that have received outstanding reviews in such magazines as Computer Language, Dr. Dobbs, and PC Tech Journal. Our compilers are currently used by industry leaders such as Ashton-Tate, AutoDesk, ANSA, and Lifetree. Now you can get them generating 80386 code.

If you have an application that requires the large 32-bit address space and the full 32-bit registers of the 80386, expand your marketplace to the rapidly growing supply of 80386 MS-DOS machines. Contact MetaWare for your 80386 software solution today! (408) 429-6382, telex 493-0879.

Durable Software Constructed Automatically™





WareTM

386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386

INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

SCIENTIFIC/ENGINEERING GRAPHIC TOOLS

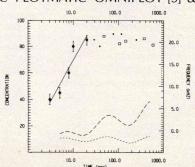
for the IBM PC and compatibles

FORTRAN/Pascal tools: **GRAFMATIC** (screen graphics) and **PLOTMATIC** (pen plotter driver)

These packages provide 2D and 3D plotting capabilities for programmers writing in a variety of FORTRAN/Pascal environments. We support MS, R-M, LAHEY FORTRAN and more. PLOTMATIC supports HP or Houston Instrument plotters. Font module available too!

Don't want to program? Just ask for **OMNIPLOT!** Menudriven, fully documented integrated scientific graphics. Write or call for complete information and ordering in structions.

GRAFMATIC-PLOTMATIC-OMNIPLOT [S] & [P]



Microcompatibles, 301 Prelude Drive, Silver Spring, MD 20901 (301) 593-0683

CIRCLE 286 ON READER SERVICE CARD

finish

End Listings

TENTION **C-PROGRAMMERS** File System Utility Libraries All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable. BTree Library High speed random and sequential access Multiple keys per data file with up to 16 million records per file. Duplicate keys, variable length data records. ISAM Driver Greatly speeds application development. Combines ease of use of database manager with flexibility of program-Supports multi key files and dynamic index definition. Very easy to use <u>59.00</u> Make Patterned after the UNIX utility. Works for programs written in every language Full macros, File name expansion and built in rules. Full Documentation and Example Programs Included.

CIRCLE 259 ON READER SERVICE CARD

1343 Stanbury Drive

Dealer inquiries invited

(416) 825-0903

Oakville, Ontario, Canada

Amazing

Your Original AMIGA™ Monthly Resource

FEATURING

- · Complete Amiga Hardware and Software reviews
- •A vast and growing library of over 110 PDS Disks
- Solid and informative for both the advanced and beginning Amiga User
- •Understandable program listings and tools
- ·Step by Step Hardware projects

Amiga Users have made Amazing Computing $^{\text{TM}}$ the longest running Monthly magazine dedicated to the Commodore Amiga. If you are searching for Amiga technical information that is both current and comprehensive, then be amazed by the pioneer Amiga Magazine, Amazing Computing - your Original AMIGA Monthly Resource.

YES, Amaze Mel I have enclosed \$24.00 U.S. (\$30.00 Canada & Mexico, \$35.00 Overseas) in check or money order (U.S. funds drawn on a U.S. bank) to:

PiM Publications, Inc. P.O. Box 869-DD Fall River, MA 02722

Fall River, MA 02722

Name _______
Address _______St_____St____Zip_

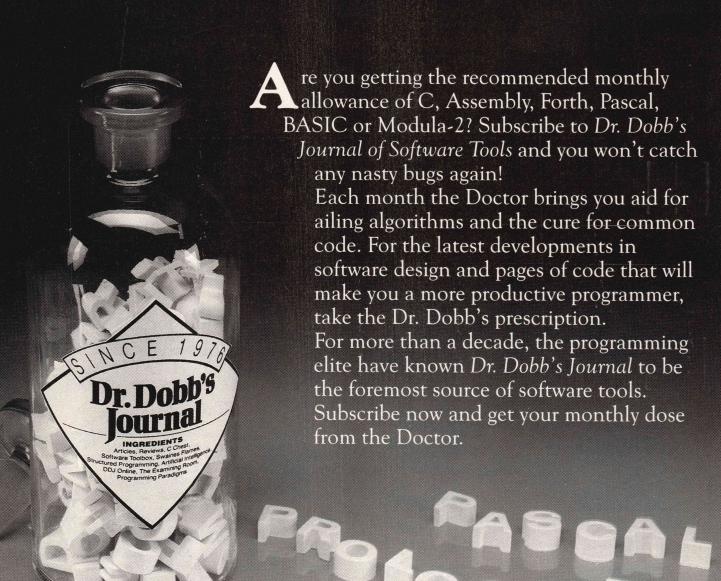
CIRCLE 136 ON READER SERVICE CARD

ALL THREE PRODUCTS FOR

For more information call or write:

Credit cards accepted

THE CURE FOR COMMON CODE



Dr. Dobb's Journal

Software Tools

The

Programmers

Subscribe Now &

Save Over 15%

Off the Newsstand Price!

SUBSCRIBE AND SAVE! Subscribe to

R. DOBB'S JOURNAL OF SOFTWARE TOOLS and save over \$5-a 15% savings off the cover price!

Please charge my: Payment enclosed ☐ Bill me later

Master Card

American Express

Card #

Signature

Name __

City_

_ State _____ Zip __

NLY \$29.97! YOU SAVE OVER \$5.00

Savings based on a full one-year cover price of \$35.40. Canada & Mexico add \$10 for surface mail per year. All other countries add \$27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue

A Publication of M & T Publishing, Inc.

UBSCRIBE AND SAVE! Subscribe to

DR. DOBB'S IOURNAL OF SOFTWARE TOOLS and save over \$5-a 15% savings off the cover price!

Please charge my:

☐ Visa

Master Card

American Express

Payment enclosed

☐ Bill me later

Card # __

_ Exp. date __

Signature ____

Name ___

City.

NLY \$29.97! YOU SAVE OVER \$5.00

Savings based on a full one-year cover price of \$35.40. Canada & Mexico add \$10 for surface mail per year. All other countries add \$27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue

A Publication of M & T Publishing, Inc.

3457

OMMENTS & SUGGESTIO

Dear Reader.

September 1987, #131

Dr. Dobb's has a long tradition of listening to its readers. We like to hear when something really helps or, for that matter, bothers you. In this hectic world of ours, however, it is often difficult to take time to write a letter. This card provides you with an easy way to correspond and, if you include your name and address, we may use appropriate comments in The Letters column. Simply fill it out and drop it in the mail.

Which articles or departments did you enjoy the most this month? Why?

Comments or suggestions _

Address:

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools

Box 3713 Escondidio, CA 92025-9843

Halaaalalllaaaalabalabalaalaalaalaalabalabala

No Postage Necessary If Mailed In The United States

Dr. Dobb's Journal of Software Tools

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools

Box 3713 Escondidio, CA 92025-9843

No Postage Necessary If Mailed In The United States



for Programmers

Subscribe Now &

Save Over 15%

Off the Newsstand Price!

PLACE STAMP HERE

Dr. Dobb's Journal of Software Tools

501 Galveston Drive Redwood City, CA 94063

Make \$500/hr.

Now develop DBMS applications 10 times faster for only \$199 with MAGIC PC — or your money back!

Database programmers, why waste your time hacking out code?

Imagine how much faster and more profitable you'd be if you could whip up powerful database applications without the time-consuming coding pains... Introducing Magic PC from Aker, your pro-

fessional dream come true. It's not another line-by-line syntax treadmill like any DBMS or 4GL.

Finally you can program as quickly as you design, while you delegate all the mundane and redundant coding tasks to Magic PC.

Program 10 times faster

Develop relational database applications 10 times faster using a visual designdriven inter-



face. Instead of writing mountains of "how to" procedural code, you quickly place your program design specs in Execution Tables and Magic PC's engine executes them automatically. Don't lose any more time editing and debugging programs by hand.

Incredible Zoom power



Magic PC's phenomenal Zoom power magically coexecutes related programs

through nested Zoom windows smoothly with auto data scrolling in all directions. While Zooming, query and transfer data across windows or even Zoom deeper.

No more maintenance!

Change your programs on the fly without any manual maintenance responsibility. Magic PC automatically updates your changes online since all the data describing your design (data dictionary, programs and menus) make up a single file, selfmaintaining Integrated Library.

Magic PC does it all

Design your entire database application with only one comprehensive development system. Generate both online programs (screens, windows, menus), as well as batch programs (reports, updates,

CIRCLE 369 ON READER SERVICE CARD

import/export, etc.) with full color and graphics. You no longer fall between the cracks dealing with separate and inconsistent programming utilities.

Free LAN features

Develop multi-user applications for local area networks with Magic PC's automatic support for file and record locking security.

Quick prototyping

Prototype a complete working application in just hours and get immediate customer feedback to finalize the design. It's a true time-saver.

Stand-alone runtime

Distribute your applications and protect your design with a low cost runtime engine. It has the friendliest end-user visual interface you've ever seen with built-in, menudriven and syntax-free data retrieval power.

Jeff Duntemann, PC Tech Journal:

"Magic PC is probably the best integrated database application generator that we have seen...very smooth system, and smoothness comes at a premium these days." Also recommended by PC Magazine, PC World, PC Week, Computer Language, Data Based Advisor and many more around the world.

Try it for \$1995

If you develop database applications for a living, you can't afford not to try Magic PC for yourself right now. For \$19.95 you'll get the Magic PC Tutorial software and documentation for hands-on evaluation, complete with a step-by-step guide to develop an Order Entry sample application in just a few hours.

Magic PC 3695 \$199

No kidding! For a limited time only, save almost \$500 off the \$695 list price, and get the complete unprotected Magic PC software for only \$199 at our special introductory non-resale price

Money back guarantee

Even at \$199 you can't go wrong with our norisk guarantee: keep it only if it makes magic for you, or we'll buy it back

within 30 days less \$19.95

restocking fee.

System Requirements: IBM PC, XT, AT, PS/2 and 100% compatible. PC-DOS 2.0 or later, 512K, hard disk. All trademarks acknowledged

MAGIC PC

The Will Database Language





City/St/Zip_

Acct. No.

Exp. Date_ Name on card Signature .

Check enclosed, or charge:

STAT Toolbox for Turbo Pascal

Bring convenience, power and versatility to your statistics programs!

Two statistical packages in one!

A library disk and reference manual

Use these powerful statistical routines to build your applications. Routines include: • statistical distribution functions • random-number generation • basic descriptive statistics • parametric and non-parametric statistical testing • bivariate linear regression, multiple and polynomial regression.

A demonstration disk and manual

This package incorporates the library of routines into a fully functioning statistical program. Two data management programs are included to facilitate the storage and maintenance of data.

Full source code is included. (For IBM PC's and compatibles. Turbo Pascal version 2.0 or later, and PC-DOS 2.0 or later are required.)

STAT Toolbox

Item #22-4 \$69.95

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me the STAT Toolbox for only \$69.95
Subtotal
CA residents add sales tax %
Add \$2.25 per item for shipping
TOTAL
Name
Address
City
StateZip
☐ Check Enclosed. Make payable to M&T Publishing. Please Charge my ☐ VISA ☐ M/C ☐ AMEX
Card No.
Exp. Date
Signature
3131B

C CHEST

Listing One (Text begins on page 106.)

```
1| #define PUBLIC
 21
 3| #ifdef DEBUG
 4 | #
            define PRIVATE
 51 #
            define D(x) x
 6| #else
 71 #
            define PRIVATE static
 81 #
            define D(x)
 91 #endif
101
11| #ifdef MSDOS
12| #
            define MS(x) x
131 #
            define UX(x)
14| #else
151 #
            define MS(x)
16| #
            define UX(x) x
17| #endif
```

End Listing One

Listing Two

```
1 /* Defines for IBM-PC Hardware */
 21
 31
   /* Timer defines:
 41
 51
     * TIMR CLK
                     The number of system clock cycles in a second
 61
                      (ie. the input frequency of the counter/timer).
     * TIMR TICKS(d) The number of ticks needed to get a delay of d
 71
 81
                     seconds. 'd' can be a fraction: TIMR TICKS (.5).
 91
101
    * TIMR_CTRL
                     Address of control port
111
     * TIMR 2 DATA
                     Address of counter 2 data port
121
     * TIMR_2_LOAD
                     Control word to load new count into
                     timer 2 (the speaker--2 bytes, 1sb first).
131
141
                     Timer initilized in mode 3 (square wave)
151
     * TIMR O LOAD
                     Same but for timer 0 (system clock).
     * TIMR 0 DATA
161
171
181
19| #define TIMR CLK
                             1193180L
20| #define TIMR TICKS(d)
                             (int) ((double) (d) * (TIMR CLK/65536.0))
211
22| #define TIMR_CTRL
                             0x43
231
241 #define TIMR 0 DATA
                             0×40
25| #define TIMR 0 LOAD
                             0x36
261
271
    #define TIMR 2 DATA
                             0x42
28| #define TIMR 2 LOAD
                             0xb6
291
30| /* Programmable peripheral interface:
311
321
            PPT
                             Base address of interface
331
            PPI SPKR
                             Bit mask to enable speaker (bit 0 is
341
                             gate on timer chip, bit 1 actually
351
                             enables the speaker).
     */
361
371
38| #define PPI
                             0x61
39| #define PPI_SPKR
                             0x03
401
411 /
42| * Make the speaker beep at a particular frequency:
431
441
    * SETFRQ(freq);
                        Sets the frequency, freq can be floating
451
461
    * SPKR ON();
                        Turns the speaker on.
    * SPKR_OFF();
471
                         Turns it off again.
481
491
50| #define SETFRQ( freq )
511
        if(1)
521
        {
531
            unsigned int count;
```

```
541
551
            count = TIMR CLK / freq ;
561
571
            outp( TIMR_CTRL , TIMR_2_LOAD
            outp(TIMR_2_DATA, count & Oxff );
581
591
            outp ( TIMR 2 DATA, (count >> 8) & 0xff );
601
611
        else
621
63| #define SPKR ON() outp(PPI, inp(PPI) | PPI SPKR)
64| #define SPKR OFF() outp( PPI, inp(PPI) & ~PPI SPKR )
```

End Listing Two

Listing Three

```
1 /* These #defines are the frequencies 12 notes of the octave
    * starting with middle C. Multiply by two to go up an octave,
    * divide by two to go down. This is an equal-tempered scale
31
 41
    * so each note is derrived by multiplying the previous note
 51
    * by the twelfth root of two. Note that there's a little
 6| * round-off error here but this error isn't audible.
71
81
9| #define TWELFTH ROOT OF TWO
                                  1.059463095
101
11| #define C4
                     (261.6256)
                                                 C 4
                     (277.1826)
                                                 C# 4
12| #define C4 SHARP
13| #define D4
                      (293.6648)
                                                 D 4
                                                         */
                                                 D# 4
14| #define D4 SHARP
                     (311.1270)
15| #define E4
                     (329.6276)
                                                         */
                      (349.2282)
                                                 F 4
16| #define F4
                     (369.9944)
                                                 F# 4
17| #define F4 SHARP
18| #define G4
                                                 G 4
                      (391.9954)
                     (415.3047)
                                                 G# 4
19| #define G4 SHARP
                                                 A 4
20| #define A4
                      (440.0000)
21| #define A4 SHARP
                      (466.1638)
                                                 B 4
22| #define B4
                      (493.8833)
```

End Listing Three

Listing Four

```
11 #include <stdio.h>
 2| #include <tools/hardware.h>
 31
 4| beep (freq, duration)
 5| double freq;
 6| double duration;
 7 | {
 81
           /* Beep the bell on the IBM-PC for the indicated time
 91
            * (which may be fractional) at the indicated frequency.
101
            * Frequencies for various notes in an equal-tempered
111
            * scale are in <tools/notes.h>.
121
131
141
           SETFRQ ( freq );
151
161
           SPKR ON();
171
18|
           delay( (double) duration );
191
201
           SPKR OFF();
21| }
221
23 | /*----
241 #ifdef MAIN
25| #include <tools/notes.h>
261
27 | main ( argc , argv )
281 char
           **argv;
291 {
301
            double atof();
311
32| #
            if O
                beep( C * 4,
331
                                   atof(argv[1]));
            endif
341 #
351
                                              (continued on next page)
```

MetaWINDOW

Power Graphics for your PC!

PC TECH JOURNAL

"Product of the Month"

"... a technological tour de force for fast PC graphics."

NO ROYALTIES!

MetaWINDOW is an advanced, high performance graphics development toolkit which bridges the gap between low-level graphic primitive libraries and pre-packaged window managers.

Unparalleled Performance!

MetaWINDOW provides an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.

- · auto-cursor tracking
- · pull-down menus
- pop-up windows
- · comprehensive
- graphic functions
- · multiple fonts



^S ^Q

Enhanced Features!

- · Display multiple bitmap or "filled-outline" fonts.
- · Face fonts for bold, italic, underline or strike-out stylings.
- Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
- Create pop-up menus, windows and icons.
- Supports IBM's new PS/2 VGA and MCGA graphics.

MetaWINDOW comes complete with langauge bindings for 16 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 40 graphics adaptors and input devices.

MetaWINDOW

Advanced Graphics Toolkit 4 disks, 3 260 page manuals - \$195*

NEW! - TurboWINDOW/C

All the features of MetaWINDOW for Borland Turbo C! Plus \$5.00 shipping and handling

TO ORDER CALL 1-800-332-1550 For information or in CA call 408-438-1550



METAGRAPHICS

269 Mount Hermon Road Scotts Valley, CA 95066

The Turbo Pascal Toolbook

Edited by Namir Clement Shammas

Make your programming easier and more powerful with the Turbo Pascal Toolbook!

You'll find:

- an extensive library of low-level routines
- external sorting and searching tools, presenting a new database routine that combines the best features of the B-tree, B+ and B++ trees
- window management, to help you create, sort and overlay windows
- artificial intelligence techniques
- mathematical expression parsers, offering two routines that convert mathematical expressions into RPN tokens
- a smart statistical regression model that searches for the best regression model to represent a given set of data.

All routine libraries and sample programs are on disk for MS-DOS systems, and over 800K of Turbo Pascal source code is included!

Turbo Pascal
Toolbook Item #25-9 \$25.95
Turbo Pascal Toolbook
with disk Item #61-5 \$45.95

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me the Turbo Pascal Toolbook for only \$25.95
Send me the Toolbook, along with the disk for only \$45.95
Subtotal
CA residents add sales tax %
Add \$2.25 per item for shipping
TOTAL
Name
Address
City
State Zip
☐ Check Enclosed. Make payable to M&T Publishing. Please Charge my ☐ VISA ☐ M/C ☐ AMEX
Card No.
Exp. Date
Signature
3131B

C CHEST

```
Listing Four
                   (Listing continued, text begins on page 106.)
361
            beep ( C ,
                            0.5);
371
            beep ( D ,
                            0.5);
            beep( E,
381
                            0.5);
391
            beep( F,
                            0.5);
401
                            0.5);
            beep ( G ,
411
            beep( A,
                            0.5):
421
                            0.5);
            beep( B,
            beep( C * 2, 0.5);
431
441 }
451 #endif
                                                   End Listing Four
Listing Five
 1 | #include <tools/hardware.h>
 2| #include <dos.h>
31
 4 | delay ( duration )
 5| double duration;
 61 {
 71
                 Delay for the indicated number of seconds (may be
 81
                 fractional.
91
             */
101
111
            unsigned long start, elapsed;
121
131
            unsigned long i, t;
141
151
            elapsed = TIMR_TICKS( duration );
161
171
            for( start = ticks(); ticks() - start < elapsed ;)</pre>
181
191 }
201
21 | ticks()
22 | {
231
        /* Return the number of BIOS clock ticks since midnight.
241
         * The routine rolls over successfully at midnight (1573040
251
         * is the number of clock ticks in a day and AL is zero
261
         * if the timer has not passed midnight since the last
         * call).
271
281
         */
291
301
        union REGS
                        regs;
311
321
        regs.h.ah = 0;
331
        int86( Oxla, &regs, &regs ); /* Time-of-day interrupt */
341
351
361
                  ( regs.h.al ? 1573040L : 0 )
371
                + ( regs.x.cx << 16
381
                    regs.x.dx
391
40| }
411
42| #ifdef MAIN
43 | main ()
441 {
451
            printf("Should be five seconds between beeps\n\007");
461
            delay(5.0);
            printf("\007");
471
48| }
49| #endif
                                                   End Listing Five
 Listing Six
       PAGE
              56,132
 21
       TITLE SPEEDUP.ASM: System-clock-modification routines
 31 ;----
 4| DEBUG equ 1 ; Set to 1 to make internal symbols public
 51 ;----
 61
```

```
TEXT
            SEGMENT BYTE PUBLIC 'CODE'
 81
    TEXT
            ENDS
 91 DATA
            SEGMENT
                     WORD PUBLIC 'DATA'
10| DATA
            ENDS
111 CONST
            SEGMENT
                     WORD PUBLIC 'CONST'
121 CONST
            ENDS
13| BSS
            SEGMENT
                     WORD PUBLIC 'BSS'
141
    BSS
            ENDS
15 | DGROUP
            GROUP
                    CONST,
                             BSS,
                                     DATA
161
            ASSUME CS: TEXT, DS: DGROUP, SS: DGROUP, ES: DGROUP
171
18| EXTRN
             chkstk:NEAR
191
201
211
221 TIMR CTRL = 43H
                       ; address of timer control port
; address of counter 0 data port
                            ; address of timer control port
23| TIMR 0 DATA = 40H
24| TIMR 0 LOAD = 36H
                            ; control word for timer
251
261 ;-----
271
    _TEXT
28|
              SEGMENT
291
301 :-----
31 ; Misc. variables. Note that I'm putting all these in the
32| ; code ( TEXT) segment so that I can find them when an
33| ; interrupt comes along. The PUBLIC statements are just for
34| ; debugging.
351
36| old int
                equ
                             ; Offset of old timer interrupt
371 old off
                dw
                     ?
                             ; service routine.
381
                             ; Segment address of same.
                    ?
39| old seg
                dw
401
                             ; User-supplied interrupt service
                     ?
41| service
                dw
421
                             ; routine (offset).
431
441 old ds
                             ; segments for running program.
                dw
 45| tick reset dw
                     2
                              ; Initialized to tick reset, decre-
 461 numticks
               dw
                             ; mented on each timer interrupt,
 471
                             ; reset to the speedup factor (and the
481
                             ; old service routine is called) when
 491
501
                             ; it reaches zero.
 511
                dw 64 dup (0); Local stack for service routine
 521 stack
                                ; 18 bytes are used by real service
 531
                                ; routine, the rest is available
 541
                                ; for the user service routine.
 551
 56| stack end
                 dw
                 dw ?
 571 old sp
 58| old ss
 591
 601 IF DEBUG
         PUBLIC old_int, old_off,
                                       old seg, old ds
 611
         PUBLIC service, tick reset, numticks, serv
 621
 631 ENDIF
 641
        ______
 651 ;-
 66| ; cli(); sti(); Disable and enable interrupts.
 671 ;
 68| PUBLIC cli, _sti
 691
             PROC NEAR
 701 _cli
 711
             cli
 721
             ret
 73| _cli
             ENDP
 741
 75| sti
             PROC NEAR
 761
             sti
 771
             ret
 781
     _sti
             ENDP
                                            (continued on next page)
 791
```

db PASCAL \$29.95

Read, write and create dBase III and compatible data files from Turbo Pascal programs. Allows dBase reporting with turbo speed and power. Accesses dBase fields using dBase field names.

db DOS \$39.95

Read and write dBase III and compatible data files from the DOS prompt. Allows fast, full screen data file editing and will query all directory files for dBase compatibility and much more.



P.O. Box 1267 Chandler, Arizona 85224-1267

Phone orders 1-800-433-6854 accept Visa/ Mastercard. For information call (602) 435-2370. Shipped by ground worldwide for \$2.50 on receipt of funds. COD extra.

CIRCLE 226 ON READER SERVICE CARD

NEW! TLIB™ 4.0 SOURCE CODE CONTROL

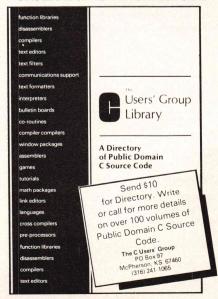
The best keeps getting better!

- · Ver. 3 reviewed in Sept 87 PC Tech Journal!
- The fastest, most powerful system is now even faster!
- Many new features! Keyword support inserts date, version, history, etc. into source code. Extended wildcard and list-of-file support; can create lists by scanning source code for includes. Branching support, for multiple development lines. Can merge (reconcile) multiple simultaneous changes.
- Keep all versions of a source code file in one compact library.
 Synchronized control of multiple related source files.
- LAN-compatible! Share libraries with all popular networks. Check-in/out locking for multi-programmer projects.
- Designed for the future! Ideal for use with WORM optical disks, like the new IBM 3363, since libraries are appended, not replaced, when you add new versions.
- Includes a copy of Landon Dyer's excellent public domain MAKE utility (with source code for DOS & VAX/VMS).
- Plus: File compare utility. Virtually unlimited source file size. Date, comments with each version. Configurable user interface, and many configurable options, like: readonly libraries, automatic tab/blank conversion, more.

PC/MS-DOS 2.x & 3.x Just \$99.95 + \$3 s/h Visa/MC

BURTON SYSTEMS SOFTWARE P. O. Box 4156, Cary, NC 27519-4156 (919) 469-3068

CIRCLE 212 ON READER SERVICE CARD



CIRCLE 181 ON READER SERVICE CARD

TURBO Advantage Display:

Form Generator for Turbo Pascal

Now, even if you have little programming knowledge, you can design and process forms to fit your needs!

TURBO Display includes a menu driven form processor, 30 Turbo Pascal procedures and functions to facilitate linking created forms to your program, and full source code and documentation. For MS-DOS systems.

Some of the TURBO Advantage: Source Code Libraries for Turbo Pascal routines are necessary to compile TURBO Display. You save \$20 when you order TURBO Advantage: Source Code Libraries for Turbo Pascal together with TURBO Display: Form Generator for Turbo Pascal! Receive both for only \$99.95!

TURBO Display: Form Generator for Turbo Pascal is also available individually for \$69.95.

TURBO Advantage/

Display Package Item TURBO Display Item

Item #070B \$99.95 Item #28-3 \$69.95

3131B

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

Display Package for only \$99.95
Send me Turbo Display: Form
Generator for Turbo Pascal for \$69.95
Subtotal
CA residents add sales tax %
Add \$2.25 per item for shipping
TOTAL
Name
Address
City
State Zip
☐ Check Enclosed. Make payable to M&T Publishing
Please Charge my □ VISA □ M/C □ AMEX
Card No.
Exp. Date

C CHEST

Listing Six (Listing continued, text begins on page 106.)

```
81| ; speedup( factor, routine )
 82| ; int factor, (*routine)();
831 :
        Speed up the system clock by the indicated factor.
 841 ;
851;
        Call the indicated subroutine on every timer interrupt.
861;
        and call the default clock routine as well every "factor"
871;
        ticks.
881;
89| ; Offsets to arguments:
901;
             factor = [bp+4]
911;
             routine = [bp+6]
921:
93| PUBLIC speedup
 941
 95| speedup
                      PROC NEAR
 961
             push
                     bp
 971
             mov
                      bp, sp
 981
             xor
                      ax, ax
 991
             call
                      chkstk
1001
1011
             mov
                      ax, [bp+6]
                                            ; service = offset of new
                      _TEXT:service,ax
1021
             mov
                                            ; routine.
1031
                       TEXT:old ds,ds
             mov
                                            ; remember current DS too.
1041
             mov
                      ax, [bp+4]
                                            ; tick reset = numticks
105|
                      TEXT:tick_reset,ax ; = ax = factor
             mov
1061
             mov
                      TEXT: numticks, ax
1071
1081
                      al, TIMR 0 LOAD
                                            ; Set up timer for load
             mov
1091
             out
                      TIMR CTRL, al
1101
             mov
                      ax, [bp+4]
                                            ; if ( factor == 1 )
1111
             cmp
                      ax,01H
112|
             jne
                      do div
                                                use 0 for the ouput count
1131
             mov
                      ax,0
1141
             jmp
                      load
                                            ; else
115| do div:
                                            ; {
1161
                      ax,00000H
             mov
                                                  Number of ticks =
1171
                      dx,00001H
             mov
                                                        65536/factor
1181
             mov
                      bx, [bp+4]
                                                  BX = factor.
1191
             div
                      bx
                                                 AX = number of ticks
120 | load:
121|
             out
                      TIMR 0 DATA, al
                                              Send new count to timer
1221
             mov
                      al, ah
123|
                      TIMR_O_DATA, al
             out
1241
1251
                                            ; Get the old vector
1261
             mov
                      ah, 35H
1271
                      al. 08H
             mov
1281
                      21H
             int
                      _TEXT:old_off,bx
1291
             mov
130|
             mov
                      TEXT:old seg,es
1311
1321
                                            ; set up the new vector
1331
             mov
                      ah, 25H
1341
                      al,08H
             mov
1351
                      dx, OFFSET TEXT: serv ;
             mov
1361
             push
                      ds
1371
             push
                      CS
1381
                      ds
             pop
1391
             int
                      21H
1401
             pop
1411
1421
             mov
                      sp, bp
1431
                     bp
             gog
1441
             ret
1451
146| _speedup
                      ENDP
1471
1481 ;-
149| ; Actual interrupt service routine. This routine saves the
150| ; environment, calles the user-supplied C service routine,
151|; and then calls the default service routine if necessary.
```

Signature

```
152| ; The service routine runs under its own stack so stack
153| ; checking should be disabled with either the /Gs command-
154| ; line switch or the "#pragma check_stack[+|-]" directive.
156| serv
             PROC
                      NEAR
1571
1581
                                                  ; Save AX on old stack
             push
1591
1601
             mov
                       TEXT:old sp, sp
                                                  ; Set up local
1611
             mov
                      TEXT:old ss,ss
                                                  ; stack
1621
             push
                      CS
163|
                      SS
             pop
                      sp, offset TEXT: stack end;
1641
             mov
1651
166|
             push
                                       ; Set up C environment:
1671
             push
                                               save everything (the
                      CX
1681
             push
                      dx
                                               flags are saved as
1691
             push
                                               part of the interrupt
                      bp
1701
             push
                                               processing) .
                      si
1711
             push
                      di
1721
             push
                      ds
1731
             push
                      es
1741
1751
             mov
                      ds, TEXT:old ds;
                                               fix the data segment
1761
             mov
                      es, TEXT:old ds;
                                               and extra segment
1771
1781
             cli
1791
                      word ptr TEXT:service ; Call C subroutine
              call
1801
             sti
1811
1821
              pop
                                               ; restore everything
183|
             pop
                      ds
                                               ; but AX
1841
             pop
                      di
1851
                      si
             pop
1861
                      bp
             pop
1871
             pop
                      dx
1881
              pop
                      CX
1891
             pop
1901
                      ss, TEXT:old_ss
1911
              mov
                                               ; Restore original
                      sp, TEXT:old sp
1921
              mov
                                                       stack.
1931
                                               ; if (--numticks > 0)
                       TEXT: numticks
1941
              dec
                      do old int
1951
              jle
                                               ; {
                      al,20h
1961
                                                     send EOI
              mov
                      20h, al
1971
              out
1981
              pop
                                                     restore ax
1991
              iret
                                               ; }
                                               ; else
2001
201| do_old_int:
                                               ; {
              mov
2021
                      ax, TEXT:tick reset
                                                    numticks
                      TEXT: numticks, ax
                                                        = tick_reset;
203|
              mov
2041
              pop
                                                     restore ax
                      dword ptr TEXT:old int ;
2051
                                                     jmp to old vector
              jmp
2061
2071 serv
              ENDP
2081
2091 :-----
2101
211| PUBLIC _slowdown
2121
                      PROC NEAR
213| slowdown
2141
2151
              push
                      bp
2161
              mov
                      bp, sp
2171
              xor
                      __chkstk
2181
              call
2191
2201
              mov
                      ax, TEXT:old_off ; See if the interrupts have
2211
              or
                      ax, ax
                                                           changed.
2221
                      no int
                                        ; No, don't fix them then
              jz
2231
2241
                                        ; restore old timer interrupt
2251
              push
                      ah, 25H
2261
              mov
                                              (continued on next page)
```



	OUTSIDE	OKLAHO	MA: NO SA	LES TAX	
E: Zenith 150, & Plus; hp Vectra	1Mbit 1Mbit 51258 4464 41256	1000Kx1 256Kx4 256Kx1 64Kx4 256Kx1	150 ns 80 ns	\$25.00 32.00 6.75 3.50 4.95	* ATIC STATION COLONIA CORRES FORFA
UPGRAD Portable	41256 41256 41256 4164	256Kx1 256Kx1 64Kx1 EPF	150 ns ROM	1.50	80387-16 ^C C \$525.00
640K MOTHERBD IBM PC/XT, Compaq	27512 27C256 27256 27128	16Kx8	200 ns 250 ns 250 ns 250 ns CRAM	\$11.25 6.50 5.50 4.75	-2 80287-8 .00 \$250.00
	43256L-12 6264LP-15 61/2 DAYS, 7:	8Kx8	120 ns 150 ns м: SHIP VIA	\$12.75 3.30 FED-EX C	SA 8087-2 \$160.00

SAT DELIVERY INCLUDED ON FED-EX ORDERS

SUNDAYS & HOLIDAYS: SHIPMENT OR DELIVERY, VIA U.S. EXPRESS MAIL MasterCard/VISA or UPS CASH COD Factory New, Prime Parts Po FED-EX ORDERS RECEIVED BY:
Th: Std Air 94/1 lb
Fr. P.1 \$10.50/2 lbs

MICROPROCESSORS UNLIMITED, INC. 24,000 S. Peoria Ave. (918) 267-4961
BEGGS OK. 74421
No minimum order. Please note that prices are subject to

CIRCLE 105 ON READER SERVICE CARD

Cross Assemblers

Universal Linker Powerful Librarian

PC/MS DOS, micro VAX. VAX VMS, VAX UNIX/ULTRIX

- Targeting over 30 MicroprocessorsVersion 2.1 is FAST
- Powerful Macros
- Absolute or Relocatable Code
- · Compatible with all Assemblers
- Conditional Assembly
- \$295 up for Completé Packages

Next Month

Microcontroller Ccross compilers



19 Jenkins Ave. Lansdale, PA 19446 U.S.A. telephone: 215-362-0966 telex: 4948709 ENERTEC

CIRCLE 346 ON READER SERVICE CARD

Dr. Dobb's Journal

Subscription Problems? No Problem!



Give us a call and we'll straighten it out. Today.

Outside California CALL TOLL FREE: 800-321-3333

Inside California CALL: 619-485-6535 or 6536

TURBO Advantage:

Source Code Libraries for Turbo Pascal

This library of more than 220 routines, complete with source code, sample programs and documentation will save you hours developing and optimizing your programs!

Routines are organized and documented under the following categories: bit manipulation, file management, MS-DOS support, sorting, string operations, arithmetic calculations, data compression, differential equations, Fourier analysis and synthesis, matrices and vectors, statistics, and much more! All source code is included.

A detailed manual includes a description of the routine, an explanation of the methods used, the calling sequence, and a simple example. For MS/PC-DOS systems.

TURBO Advantage: Source Code Libraries for Turbo Pascal is also available with TURBO Advantage Complex: Complex Number Routines for Turbo Pascal and TURBO Advantage Display: Form Generator for Turbo Pascal. See pages and

Turbo Advantage Item #26-7 \$49.95

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me the Turbo Advantage:
Source Code Libraries for Turbo
Pascal for only \$49.95
Subtotal
CA residents add sales tax %
Add \$2.25 per item for shipping
TOTAL
Name
Address
City
State Zip
☐ Check Enclosed. Make payable to M&T Publishing
Please Charge my \square VISA \square M/C \square AMEX
Card No.
Exp. Date
Signature
3131B

C CHEST

```
Listing Six (Listing continued, text begins on page 106.)
2271
                    al,08H
            mov
2281
                    ds,_TEXT:old seg;
            mov
2291
            mov
                    dx, TEXT:old off;
2301
                    21H
            int
2311
            pop
2321
233| no int:
2341
            mov al, TIMR 0 LOAD ; Restore default system
2351
            out TIMR CTRL, al
                                    ; clock tick rate
236|
            mov
                 al,0
2371
            out
                 TIMR O DATA, al
2381
            out TIMR 0 DATA, al
2391
2401
                    sp, bp
            mov
2411
            pop
                    bp
2421
            ret
2431
244| _slowdown
                    ENDP
2451
246| TEXT
          ENDS
247 | END
                                                   End Listing Six
 Listing Seven
 1| #include <stdio.h>
 2| #include <signal.h>
 3| #include <stdarg.h>
  4| #include <ctype.h>
 5| #include <tools/notes.h>
                                   /* Frequencies of notes */
  6| #include <tools/hardware.h>
                                 /* TIMR CLK
                                                            */
  7| #include <tools/debug.h>
                                   /* D() macro
                                                            */
 81
 91 /* CLICK.C
                    A polyrhythmic metronome. Usage is described
 10| *
                    in the usage msg[], below.
 111
 121
                    (c) 1987, Allen I. Holub. All rights reserved.
 131
 14| */
 151
 16| extern double ceil ( double );
 17| extern double
                   floor ( double );
 181
 19| /*-----
 20| * The compiler truncates floating point numbers when they're
 21| * converted to int. This macro rounds as it converts.
 221
 231
 24| #define ROUND(x) ((int)(((x) > 0.5) ? ceil ((double)(x)) \
 251
                                         : floor((double)(x))))
 261
 271 /*----
 28| * TICKS(x) converts a metronome count to clock ticks.
 291
 30| * With a speedup factor 4, a tick happens 72.84 times/second
     * (every .01373 seconds, more or less). A speedup factor of
 311
     * 2 yields half this number: 36.42 times/sec, or an interrupt
 321
 33| * every .02746 seconds).
 341
 351
     * A metronome 60 is 1 Hz, 120 is 2 Hz, etc.
 361
 371
     * seconds
                             60 / metronome count
     * ticks in second == ( 60 / metronome_count ) * ONE_TICK
 381
                        == ( (60 * ONE TICK) / metronome_count )
 391
 401 */
 411
                                                  /* Speedup factor */
 42| #define FACTOR
                            (TIMR CLK / 65536.0) /* ticks / second */
 43| #define DEFAULT TICK
 44| #define ONE TICK
                            (DEFAULT TICK * FACTOR)
 451
                             ROUND ( (60.0 * ONE TICK) / (x) )
 46| #define TICKS(x)
 471
```

```
481 /*----
491
50| #define min(a,b)
                             ((a) < (b) ? (a) : (b))
51| #define max(a,b)
                             ((a) > (b) ? (a) : (b))
521
53| #define MAX MEASURES
                            1280
                                     /* Max # of measures/track
                                                                  */
54| #define NUM_TRACKS
                                     /* Number of tracks
551
56| #define WARNING
                             (NUM TRACKS + 1)
571
581 /*-----
591
60| typedef unsigned char
                           uchar;
61| typedef unsigned int
                            uint;
621
63| typedef struct
64 | {
                               /* # of beats remaining in measure */
        uchar num beats;
651
        uchar remainder;
                               /* Number of ticks to get in synch */
661
                               /* Current tick for this beat
                                                                   */
671
        uint cur tick;
        uint ticks per beat ; /* # of clock ticks between beats
                                                                   */
681
        uint metronome: 14; /* metronome count in this measure */
691
        uint silent : 1 ; /* this measure is silent
                                                                   */
701
        uint warning : 1 ; /* warning tone at downbeat
711
72| }
73| MEASURE;
741
75| typedef MEASURE TRACK [ MAX MEASURES ];
761
771
    TRACK
                      [ NUM TRACKS ];
            Tape
78 | MEASURE *Measure [ NUM TRACKS ];
                                      /* Current measure on */
                                       /* each track of Tape.*/
791
                                       /* Input line number */
801 int
            Lineno = 0:
811
            Ring_bell = 0; /* 0 if the bell shouldn't ring.
821 int.
831
                              * Set to 1 for track 1, 2 for track
841
                              * 2, etc.
851
861
            Collision = 0; /* If two track collide, the track
87| int
                              * number of the second one is
881
891
                              * put here.
901
911
                             /* Incremented by the interrupt
921 int.
            Downbeat = 1:
931
                              * service routine on every
941
                              * downbeat from track 0;
951
961
                             /* Set to 1 by interrupt service
971 int
             Done = 0;
                              * routine when it gets to the
981
                              * end of the tape.
991
1001
101|
                             /* For debugging, incremented on
1021 int.
             Numticks = 0;
                              * every timer interrupt.
103|
1041
105|
1061 /
1071
108| char
             *Usage msg[] =
109| {
         "Usage: click [-d] inputfile",
1101
         "",
1111
         " -d (for dull) don't use different notes for different",
1121
1131
              tracks",
1141
         11 11
         "A polyrhythmic metronome. Four independent \"tracks\"",
115
         "are supported, with rhythms specified as follows:",
116|
1171
         "track 0: #4 @120 5/8, #6 @120 6/8, @120 4/4",
1181
         "track 1: #4 6 , #6 @100 6/8",
1191
         1111
1201
                                            (continued on next page)
```

We supply tools for the most advanced systems programming language in widespread use:

MODULA-2

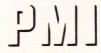
Our Products Include:

- ★ EmsStorage™: high-level storage module that detects and uses LIM Expanded Memory if present, or DOS memory if not; lets users conserve scarce DOS memory for other programs. Fast allocation & deallocation. Automatic garbage collection. Provides MS-Windows-like interface (lock/unlock functions) for porting programs to Windows and OS/2.......\$49
- ★ Graphix: the authorized Modula-2 interface to MetaWINDOW, the professional graphics system PCTJ named 7/85 Product of the Month; includes full MetaWINDOW package.... \$149
- ★ Macro2: a macro preprocessor for Modula-2; provides inline expansion of functions, include files, conditional compilation, etc.
 With full source: \$89

★ Full source code for *Make* and *Xref* utilities; customize them to work exactly as you like.

Each: \$89

All available exclusively from PMI; dealer inquiries welcome. Full documentation for these and many other products available on free demo disks.



Portland, OR 97206

4536 SE 50th

VISA/MC AMEX/COD/PO

(503) 777-8844

BIX: pmi CIS: 74706,262

CIRCLE 239 ON READER SERVICE CARD

TURBO Advantage Complex:

Complex Number Routines for Turbo Pascal

W orking with complex numbers is easy with the Turbo Pascal procedures and routines provided in TURBO Advantage Complex!

TURBO Complex provides procedures for performing all the arithmetic operations and necessary real functions with complex numbers. Each procedure is based on predefined constants and types. By using these declarations the size of arrays are easily adapted. Each type declaration is a record with both a real and imaginary part. Use these procedures to build more sophisticated functions in your own programs.

TURBO Complex also demonstrates the usage of these procedures in routines for vector and matrix calculation with complex numbers and variables; simultaneous Fourier transforms; calculations of convolution and correlation functions; low-pass, high-pass, band- pass and band-rejection digital filters; and solving linear boundary-value problems.

Source code and documentation is included. For MS-DOS systems. Some of the *TURBO Complex* routines are most effectively used with routines contained in *TURBO Advantage*. Receive both *TURBO Advantage* and *TURBO Complex*, together for only \$115! You save \$25!

TURBO Complex: Complex Number Routines for Turbo Pascal is also available individually for \$89.95.

TURBO Advantage/

Complex Package TURBO Complex Item #070A \$115 Item #27-5 \$89.95

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

YES! Please send me the Turbo Advan Complex Package for only \$115	ntage/
Send me Turbo Complex: Comple Number Routines for \$89.95	x
Si	ubtotal
CA residents add sales tax	%
Add \$2.25 per item for sh	ipping
Name	
Address	
City	
State Zip	
□ Check Enclosed. Make payable to M Please Charge my □ VISA □ M/C	M&T Publishing. □ AMEX
Card No.	1000
Exp. Date	
Signature	
	3131B

C CHEST

Listing Seven (Listing continued, text begins on page 106.)

```
1211
         "No line can be longer than 132 characters, but several",
1221
         "track specifiers can be given. The basic notation is:",
1231
         "\"[#N] [@Z] X[/Y],\" interpreted as N measures of X/Y at",
         "metronome Z. If the \"#N\" is missing, 1 is used. If",
1241
1251
         "the \"@Z\" is missing, the measure is stretched to",
         "synch with track one on the down beat. The \"/Y\" is",
1261
1271
         "optional. So, in the above example, the six beats in",
         "the first four measures of track 2 will synch up with",
1281
1291
         "track one, coming into synch on the downbeat of each",
1301
         "measure. Each track may be up to 1000 measures long.",
1311
         "Lines that don't begin with \"track\" are ignored",
1321
         NUIT.I.
133| };
1341
1351
1361
1371 char
             *skipwhite(p)
138 | char
             *p;
139| {
1401
         /* Skip all characters that aren't part of a command */
1411
1421
         while( *p && (isspace(*p) || *p == '\n') )
1431
                 p++ ;
1441
1451
         return p;
146| }
1471
148| /*----
1491
150| err ( fmt )
151| char
             *fmt:
152| {
              /* Works like printf() but writes to standard error and
1531
1541
              * prints an input line number along with the message.
1551
1561
1571
             va list
                        args;
158|
             va start ( args, fmt );
1591
1601
             fprintf ( stderr, "line %d: ", Lineno );
1611
             vfprintf( stderr, fmt, args );
1621 }
1631
1641 /*-
1651
166| init()
167| {
168|
             /* Initialize The Measure array to point
1691
              * at the first measure of each track on the Tape.
1701
1711
1721
             register int i;
1731
1741
             for ( i = NUM TRACKS; --i >= 0; )
1751
                     Measure[i] = Tape[i] ;
176| }
1771
1781
1791
180| print_tape ( this_many )
181 |
1821
         /* Print out the tape. If this many is 0, only the
1831
          * initialized measures are printed; otherwise, the
1841
          * indicated number of measures are printed
1851
1861
1871
         MEASURE
                       *p ;
1881
         register int i , measure num ;
1891
                       amt;
1901
1911
         for( i = 0; i < NUM TRACKS ; i++ )
1921
1931
             printf( "Track %d:\n", i );
```

```
1941
             measure num = 0;
1951
                          = this_many;
1961
1971
             for( p=Tape[i]; p->num_beats>0 || --amt >= 0; p++)
1981
1991
                 printf("
                            measure %2d: ", ++measure num
2001
                 printf("[%2d ticks/beat " , p->ticks_per_beat );
2011
                 printf("+ %d], "
                                            , p->remainder
                                                              );
                                            , p->cur tick
2021
                 printf ("cur tick=%d, "
                                                                 );
2031
                 printf( "%d beats "
                                             , p->num beats
                                                                 );
2041
2051
                  if (p->metronome)
2061
                      printf( "at %-5d ", p->metronome );
2071
2081
                      printf("in synch");
2091
2101
                 printf("%s", p->silent ? "(mute)": "");
2111
                 printf("%s", p->warning ? "(warn)" : "" );
                 printf("\n");
2121
213|
             }
2141
         }
215| }
2161
2171
2181
219| build tracks (file name)
220 | char
             *file name;
221| {
2221
         FILE
                      *fp;
2231
         MEASURE
                      *mp;
2241
                      buf[133], *line;
         char
2251
         int
                      i:
2261
          int
                                       /* Track number
                      num measures; /* # of measures to repeat
2271
         int
2281
                      metronome;
                                      /* metronome count
                                                                   */
          int
                                                                   */
2291
                      beats;
                                      /* beats per measure
          int
                                                                   */
                                      /* ticks per measure
2301
          int
                      ticks;
2311
                                      /* Current measure number
                                                                   */
          int
                      measure:
                                      /* measure is silent
                                                                   */
232|
                      silent;
          int
2331
                      warning;
                                       /* warning on last repeat
2341
2351
          if(! (fp = fopen(file name, "r")))
2361
                  return 0;
2371
2381
          init();
2391
2401
          while ( line = fgets (buf, 133, fp) )
2411
242|
              ++Lineno;
2431
2441
              line = skipwhite( line );
2451
              if( !( line[0]=='t' && line[1]=='r'
2461
2471
                    && line[2] == 'a' && line[3] == 'c' && line[4] == 'k'
2481
                   )
              ) continue;
2491
2501
                                                /* Get track number */
2511
              line += 5;
2521
              line = skipwhite ( line );
2531
              track = stoi ( &line );
              line = skipwhite ( line );
2541
2551
              if( *line == ',' || *line == ':')
2561
2571
                       line++;
2581
                                  [ track ]; /* starting measure # */
 2591
              measure = mp - Tape [ track ];
 2601
 2611
              while ( *line )
 2621
 2631
 2641
                  num measures = 1;
                                = 0:
 2651
                  metronome
 2661
                   silent
                                = 0;
                                = 0;
 2671
                   warning
 268|
                                            (continued on next page)
```

Dbase*

programming tools *Clipper, FoxBASE+, dBASE, QuickSilver

The UI Programmer

UI is the first professional code generator; we wrote UI for programmers who want to automate their work but cannot use code that is 'almost' good enough. If your user interfaces include bounce-bar menus, pop-up help screens and the other features of today's best programs, you will gain an order of magnitude in productivity with UI.

UI is a second generation, programmable product — so your code comes out your way. Application specific edits, for instance, can be placed in the UI 'template' which controls the generation. Edit the screen appearance until it 'looks and feels' perfect. Everytime you generate code, your special logic is preserved.

Speaking of editing the screen, UI includes a powerful, 3–D screen editor, so you can draw pop-up help boxes over your pull-down menus, over your application.

The Documentor

To run Doc, you just tell it the name of the mainline routine and make sure your printer has a lot of paper! (Sure, you can have the output go to the screen or a file, too.)

You can tailor your documentation to include any or all of: a table of contents, system tree diagram (main line is the root), hierarchy (box diagram) charts for each module, action diagrams (modern style flow charts) for each PRG or procedure, DBF listings (structure, indexes, more), where used/updated listings for fields and all variables — by module and by line number within each module.

Our written money-back satisfaction guarantee set a new standard when we began it in 1985. (Return rate to date: 0.6% and dropping!) No copy protection, royalties or other nonsense.

Suggested retail: \$295 each, (800) support included. At your dealer today. Call us for a very special offer on our latest release! (800) 233-3569 or, in NY, (212) 406-7026.

WallSoft

The Computer Aided Software
Engineering Corporation
233 Broadway, Suite 869, New York, NY 10279
CIRCLE 90 ON READER SERVICE CARD

C CHEST

Listing Seven (Listing continued, text begins on page 106.)

```
2691
                 for( line=skipwhite(line); *line; line=skipwhite(line) )
2701
2711
                      if( *line == ';')
                                                       /* comment
2721
                      1
2731
                              *line = 0;
2741
                              break;
2751
2761
                      else if( *line == ',' || *line == ':' )
2771
2781
                              ++line;
                                                       /* end of measure */
2791
                              break;
2801
2811
                      if( *line == '#')
                                                       /* # of measures */
2821
                     {
2831
                                      = skipwhite ( ++line );
                          num measures = stoi
2841
                                               ( &line );
2851
2861
                      else if( *line == '@')
                                                      /* metronome count */
2871
2881
                          line
                                  = skipwhite ( ++line );
2891
                          metronome = stoi
                                                ( &line );
2901
2911
                      else if( *line == 'w' || *line == 'W' )
2921
2931
                          while ( isalpha (*line ) )
2941
                              ++line:
2951
2961
                          warning = 1;
2971
2981
                      else if ( isdigit ( *line ) )
                                                      /* Time signature */
2991
3001
                          if( (beats = stoi(&line)) == 0 )
3011
3021
                              err( "Illegal time signature\n" );
3031
                              exit(1);
3041
3051
                                                       /* Throw away bottom */
3061
                          if( *line == '/')
3071
                              ++line;
                                                       /* of time signature. */
3081
3091
                          while ( isdigit ( *line ) )
3101
                              line++:
311|
                      else if( *line == '(' || *line == ')')
3121
3131
3141
                          ++ line
3151
                          silent = 1;
3161
                      1
3171
3181
                          err("<%c> is illegal in measure spec.\n", *line );
3191
3201
3211
                  for(; --num measures >= 0; mp++, measure++)
3221
                  {
3231
                      if ( metronome )
3241
3251
                          mp->metronome = metronome;
3261
                                    = TICKS (metronome) * beats ;
                          ticks
3271
3281
                      else
3291
3301
                          mp->metronome = 0;
                          ticks = Tape[0][measure].ticks_per_beat
3311
                                * Tape[0][measure].num_beats
3321
3331
3341
3351
3361
                      if ( num measures == 0 ) /* last in series */
3371
                              mp->warning = warning;
3381
339
                      mp->silent
                                         = silent ;
3401
                                        = beats
                      mp->num_beats
3411
                      mp->cur_tick
                                         = ticks / beats;
```

```
3421
                      mp->ticks_per_beat = ticks / beats;
3431
                      mp->remainder
                                           = ticks % beats;
3441
345|
                      D( printf("loading track %d, ", track
                                                                      );)
                      D( printf("%d beats/measure ", beats
D( printf("%t retree ", beats
3461
                                                                      );)
3471
                                                                      ):)
                      D( printf("at metronome %d\n" , mp->metronome);)
3481
3491
3501
3511
                  Measure[ track ] = mp;
3521
              }
3531
         }
3541
355|
          init();
         D(print_tape(0);)
3561
3571
          return 1;
3581 }
3591
360| /*----
3611
362| #pragma check_stack- /* Turn off stack probes. This pragma */
                               /* is Microsoft-compiler dependent.
3631
3641
365| timer_intr()
366| {
          /* Interrupt service routine for timer interrupt */
3671
3681
         MEASURE **mp, *p;
3691
3701
                      i, did nothing;
          int
3711
3721
          Done = 1:
3731
          ++ Numticks;
3741
3751
          for( i = 0, mp = Measure; ++i <= NUM TRACKS; mp++)
3761
3771
              if (p = *mp) \rightarrow num beats > 0)
3781
                  if( p->cur_tick==p->ticks_per_beat )
3791
3801
3811
                       /* Ring bell on first tick of measure
3821
                        * unless this is a silent measure.
                        * Warnings take precedence over
3831
3841
                           everything.
3851
3861
3871
                       if (p->warning)
3881
3891
                               Ring bell = WARNING ;
                               p->warning = 0;
3901
3911
3921
3931
                       else if ( !p->silent )
3941
 3951
                           if (!Ring bell)
                               Ring_bell = i;
 3961
 3971
                           else
                                Collision = i;
 3981
 3991
 4001
                   }
 4011
                   if ( -- p->cur_tick <= 0 )
 4021
 4031
                       if( -- p->num_beats <= 0 )
 4041
 4051
                            if( i == 1)
 4061
 4071
                                ++Downbeat;
 4081
                           ++( *mp ); /* go to next measure */
 4091
 410|
                       else
 4111
 4121
                           p->cur_tick = p->ticks_per_beat;
 4131
 4141
                            if (p->remainder > 0 )
 4151
 4161
                                             (continued on next page)
```

GenView

3 D GRAPHICS ANIMATION

Now you can have a 3 D animating graphics system for your EGA or CGA equipped PC or PC compatible. (DOS 2.1 or higher.)

- 1 Includes all source code (C and ASM)
- 2 Hidden surface removal, translation, scaling and rotation
- **3** Scripting for viewpoint and object movements
- 4 Slideshow or animated playback of display frames
- 5 Software interface to single frame camera
- 6 "Fly through" of scenes with moving objects
- 7 Variable order spline curve generation for object and view point path generation
- 8 Object description includes MACRO capability
- 9 Source code includes direct to hardware assembly routines for high speed display
- **10** Control viewpoint path, acceleration, and velocity
- **11** Control object path, velocity, and rotation

GenView allows you to specify a viewpoint path with a small set of "turning points." It will then generate a smoothed series of frame locations along the path according to the velocity and acceleration parameters you specify. View direction may be backwards, forwards, or user specified. Object movement is controlled through a similar process with the addition of rotation (about any or all axes).

CGA Animation Sequence Demo \$7.00

EGA Animation Sequence Demo \$9.00 (2 diskettes) CGA Flythrough Sequence Demo \$9.00 (Requires hard disk)

GenView System \$99.00

The GenView System includes executable code, source, MAKE files, object description library, and manual (more than 40 pages) on diskette.

No Purchase Orders. Massachusetts residents add 5% sales tax. Outside USA add \$15.00. VISA and MASTERCARD accepted.

Call 617-528-4280 to order or send check or money order to:

GenSoft Systems
Dept. 9D
P.O. Box 1
Foxborough, MA 02035

CIRCLE 166 ON READER SERVICE CARD

C CHEST

```
Listing Seven (Listing continued, text begins on page 106.)
4171
                             -- p->remainder;
4181
                            ++ p->cur tick ;
4191
                         }
4201
4211
4221
                Done = 0;
4231
            }
4241
        }
425| }
4261
427| #pragma check_stack+
428!
429| /*----
4301
431| on break()
432 | {
4331
            /* Routine for signal(), tries to put the clock rate
             * back to normal on a Ctrl-Break. Note that this
4341
              * routine can fail if Ctrl-Break is hit several times
4361
             * in quick succession.
4371
4381
4391
           signal ( SIGINT, SIG IGN );
4401
        slowdown();
4411
            exit (0);
442| }
4431
4441
                   -----*/
4451
446| print_stats()
4471 {
448| }
4491
4501 /*-
4511
452| usage()
4531 {
4541
            for(p = Usage_msg; *p; fprintf(stderr,"%s\n", *p++) )
4551
4561
4571
             exit(1);
458| }
4591
460!
4611
462 | main ( argc, argv )
463| char
            **argv;
464 | {
4651
             static int measure = 0; /* current measure in track 0 */
             static int dull = 0; /* Dull output
static int stats = 0; /* Statistics only
4661
4671
4681
             static int i;
4691
4701
             if ( argc == 3 )
4711
             1
4721
                 --argc;
4731
4741
               if( **(++argv) != '-')
4751
                   usage();
4761
4771
                 switch( argv[0][1])
4781
4791
                 case 'd': dull = 1;
4801
                 default:
                                             usage();
4811
4821
4831
             else if( argc != 2 || *argv[1] == '-')
4841
                     usage();
4851
4861
             if (!build tracks (argv[1]))
4871
                   exit(2);
488
4891
             i ( stats )
```

```
4901
4911
                      print stats();
4921
                      exit (0);
4931
4941
              signal ( SIGINT, on break );
4951
4961
4971
              for( speedup(FACTOR, timer_intr); !Done; )
4981
                                        /* Interrupts off
4991
5001
                  if ( Ring bell == WARNING )
5011
5021
5031
                      Collision = 0;
5041
                      Ring bell = 0;
5051
                      sti();
5061
                      beep ( C4*8, 0.125 );
                                     0.1 );
5071
                      delay(
5081
                      beep ( C4*8, 0.125 );
5091
                  else if ( Collision )
5101
5111
5121
                      /* the higher track wins */
5131
5141
                      i = max( Collision, Ring bell );
5151
                      Collision = 0;
5161
                      Ring bell = 0;
5171
                      sti();
                      beep ( dull ? C4*2 : C4 * i, 0.125 );
5181
5191
5201
                  else if ( Ring_bell )
5211
5221
                       /* You must set Ring bell to
                        * 0 before calling beep so that
5231
5241
                        * a note won't collide with itself.
5251
                                                                                             (continued on next page)
```

DDJ Back Issues

Rebruary 1986 #112 Volume XI Issue 2 Structured code in Pascal, Modula-2, Ada— The Great CRC Mystery: Solved—Data Abstraction with Modula-2—A Shell for MS DOS

March 1986 #113 Volume XI, Issue 3
Parallel Processing—Concurrency and Turbo Pascal—What Makes DOS Fast—Minimizing Arbitrary Functions—MC68000 vs. NS32000.

Aug. 1986 #118 Volume XI, Issue 8
Special C Issue—Benchmarking C Compilers—
The Joy of Conciseness—Nearly Perfect Trees—
Generics in Ada—Real-World Data Types.

Sept. 1986 #119 Volume XI, Issue 9 Smooth Algorithms—MS-DOS Directory Traversal—Turbo Boards Review—Radix Sort— Does Turbo Prolog Measure Up—Crawling Memory Test

Oct. 1986 #120 Volume XI, Issue 10 80386 Programming—MS-DOS File Browsing— Converting to the 320xx—Modula-2 Compiler Review—Factoring in Forth.

Nov. 1986 #121 Volume XI, Issue 11 Graphics Routines—The New Graphics Chips— Programming Tips in C, Modula-2, Pascal, and Ada—68k Graphics. Dec. 1986 #122 Volume XI, Issue 12
Multitasking—32000 Assembler—Comparing
String Comparisons—Turbo Pascal Procedural
Parameters.

Jan. 1987 #123 Volume XII, Issue 1 Annual 68K Issue, 68K Mini Forth, OS-9 Operating System, Mac and Amiga Interface Programming.

Feb. 1987 #124 Volume XII, Issue 2 Editors and Assemblers.

Mar. 1987 #125 Volume XII, Issue 3 Data Compression Techniques

May 1987 #127 Volume XII Issue 5 Notes on Computer Music—Scientific Programming—Command Processors

June 1987 #128 Volume XII Issue 6 Handling Large Priority Queues—TSR Serial Drivers—Unix Shell Scripts

July 1987 #129 Volume XII Issue 7 386 Development Tools—Optimizing 8088 Code-Curses for MS-DOS

Aug 1987 #130 Volume XII Issue 8
Unveiling ANSI C—New Tools for C—Ray Duncan
on Dos 3.3—AI: Programming in Loops
Other issues are also available. Please inquire.

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Galveston Dr. Redwood City, CA 94063. Or, call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

Please send the issues circled: 114 112 113 118 119 120 121 122 123 124 127 129 128 130 Price: 1 issue-\$5.00. 2-5 issues-\$4.50 each. 6 or more-\$4.00 each. (There is a \$10.00 minimum for charge orders.) Subtotal CA residents add sales tax Outside U.S., add \$.50 per issue ___

Name _

Signature _

Card No. _____

3131E

C CHEST

Listing Seven (Listing continued, text begins on page 106.)

```
5261
5271
                      i = Ring bell;
5281
                      Ring_bell = 0;
5291
                      sti();
                      beep( dull ? C4*2 : C4 * i, 0.125 );
5301
5311
5321
                  else
5331
                  {
5341
                           sti();
5351
5361
5371
                  if ( Downbeat != measure )
5381
                       printf( "\r%d", measure = Downbeat );
5391
5401
5411
              printf( "\rDone" );
5421
5431
5441
              D( printf("%d ticks overall\n", Numticks ); )
5451
              slowdown();
5461
              printf("\n");
547| }
```

End Listings

MACH 2

INTERACTIVE ASSEMBLY AND FORTH DEVELOPMENT SYSTEM FOR 68000/68020/68881 PROCESSORS



Mach2 for Industrial OS-9/68000

Also available for: Industrial Boards and Macintosh

Full OS-9 modular programming support.

Easy generation of program and trap modules.

Standard infix 68000/68020/68881 assembler.

Fast subroutine-threaded Forth 83 implementation.

For more information, call or write today:

Palo Alto Shipping Company
P.O. Box 7430 • Menlo Park, CA 94026
(415) 854-7994 • (800) 44FORTH

CANADA'S SOURCE FOR C

- Canadian Sales
 - Canadian Service
 - Canadian Technical Support
 - Canadian Product Knowledge

We specialize in programming & development software

LIFEBOAT • LATTICE • GREENLEAF • PHOENIX

SOFTCRAFT . MICROSOFT . BLAISE . ESSENTIAL

AGE OF REASON - DESMET - AZTEC

MARK WILLIAMS . GIMPEL . ROUNDHILL . GSS

HALO . FAIRCOM . RAIMA . INTEL . etc. . etc. .



Call for full price list—Dealer enquiries welcome



We know our products—we use them!

SCANTEL SYSTEMS LTD.

801 York Mills Rd., Don Mills, Ont., M3B 1X7 (416) 449-9252

CIRCLE 76 ON READER SERVICE CARD

The Software Family

649 Mission Street San Francisco, CA 94105

(1)800-443-7176

In California or outside U.S.

(415)583-4166

Trust TSF to provide the best value and service!

- Technical advice and support by programmers
- Honest and equitable business practices
- Prompt, flexible service to meet your needs

We accept checks, Visa, MasterCard, American Express and COD. We charge your card only when we ship and do **not** add a surcharge. We provide free UPS delivery on software orders over \$100 (\$3 delivery charge on orders under \$100). We add actual charges for UPS Blue Label or Federal Express rush service. Our COD fee is \$2. We allow return privileges on most products. We carry a large inventory and ship promptly, so you receive your shipment within 3 to 6 working days of placing your order. Give us a try.

Turbo C Specials

Borland Turbo C (list \$99).....\$59

Blaise Turbo C Tools (list \$129)\$95 ISR/TSR support, critical error trapping, direct video access, more
Creative Vitamin C (list \$225)
Gimpel PC Lint (list \$139)
Metagraphics Turbo Window/C (\$95)\$75 Graphics windows, menus and multiple fonts
Softfocus Btree & ISAM (list \$115)\$99

Basic C Turbo Pascal dBase Publishing Al

Aldebaran Source Print (list \$75)	\$59
Blaise C Tools + (list\$175)	. \$125
Blaise Light Tools For Datalight (list \$100)	\$79
Blaise Turbo Power Tools + (list \$99)	\$79
Borland Turbo Basic (list \$99)	
Borland Turbo C (list \$99)	\$59
Bytel Genifer dbase III application generator (\$395)	.\$225
Comtel dbScope Interactive dBase debugger (\$70)	\$62
Comtel dbTools (list \$70)	\$62
Creative Vitamin C Specify compiler (list \$225)	. \$150
Creative Programming VC Screen (list \$100)	\$79
Datalight Optimum C w/Easy Edit (list \$140)	\$99
Fox FoxBase + fast dBase compiler (\$395)	
Gimpel C-Terp Specify compiler (list \$300)	. \$224
Gimpel PC-Lint (list \$139)	\$95
Greenleaf Data Windows Specify compiler (\$225)	\$157
Guidelines C++ for Microsoft C (list \$195)	\$175
Lattice C (list \$500)	\$270
Matrix Synergy Development Toolkit (list \$395)	\$309
MicroHelp Mach2 for Microsoft Basic (\$69) windows, fast i/o, critical error trapping, more	\$55
MicroHelp Mach2 for Turbo Basic (\$69)	955
MicroHelp Mach2 for Turbo Pascal (\$69)	\$33 \$55
Microsoft C v5 w/Codeview & Quick C (list \$450)	\$270
Microsoft Quick C (list \$00)	\$69
Microsoft Quick C (list \$99) Microsoft Fortran w/Codeview (list \$450)	\$270
Microsoft Quick Basic v3 (list \$99)	\$65
MKS Toolkit Korn shell, vi, grep, 30 more (\$139)	\$109
Periscope Periscope I (list \$345)	\$279
Periscope Periscope II (list \$175)	
Periscope Periscope III (list \$995)	
Periscope Periscope III 10Mhz (list \$1095)	\$876
Phoenix PFix-86+ (list \$395)	\$250
Phoenix Plink-86+ (list \$495)	\$315
Tom Rettig Library (list \$99)	\$87
Turbo Power Extender (list \$85)	\$68
Turbo Power Optimizer (list \$75)	\$59
Turbo Power Optimizer w/Source (list \$175)	\$99
Turbo Power TDebug Plus v2 (list \$60)	\$48
Turbo Power Turbo Plus v2 (list \$60)	\$76
Unipress Phact RDBMS w/ report & query (\$249)	\$199
Wallsoft UI Programmer (list \$295)	\$260
Wallsoft UI Programmer (list \$295)	\$110

TSF carries hundreds of products from dozens of publishers. Call or write for a <u>FREE</u> catalog or a price quote.

Complete multi-key ISAM file system with source

MULTI USER BBS

Off-the-shelf and custom systems for:

- ★ Multi-User Teleconferencing
- ★ Multi-User Electronic Mail
- ★ Multi-User File Upload/Download
- ★ Multi-User Order Entry
- ★ Multi-User Games and Amusements
- ★ Multi-User Database Lookup
- ★ Multi-User Online Expert Systems
- ★ Multi-User Catalog Scanning
- ★ Multi-User Classified Advertising
- ★ Multi-User Educational Services



What do you need for your Multi-User Bulletin Board System?

	Us	Them
16 modems on one card	YES	?
Up to 64-user capability	YES	7
Runs under MS-DOS V3.1	YES	?
C source code available	YES	?
Menu-oriented operation	YES	?
Accounting w/audit-trail	YES	?
Extensive SYSOP displays	YES	7
Powerfail-protected data	YES	7
"Midnite cleanup" option	YES	7
1-year hardware warranty	YES	7

We sell hardware and software for the IBM PC family and compatibles. Our product line is centered around the GALACTICOMM BREAKTHROUGH, a single-slot card with 16 independent modems on it. You will simply have a cable coming out the back of your machine, going straight into the jacks in the wall installed by the telephone company. No external hardware needed.

Call our multi-user demo system with your modem, at (305) 922-3901. Then call (305) 472-9560, voice, for more information. Why not call right now?

GALACTICOMM

GALACTICOMM, Inc., 11360 Tara Drive, Plantation, FL 33325 CIRCLE 362 ON READER SERVICE CARD

STRUCTURED PROGRAMMING

```
Listing One (Text begins on page 122.)
PROGRAM CBTree;
  Program to test the compare the time needed in creating and
   searching a binary tree and a CB-tree.
  Author: Namir Clement Shammas
CONST SIZE = 1000;
      RANGE = 10000;
      MainLoopCount = 100;
TYPE Bin Ptr = 'Bin Node;
     { normal binary tree strcutures }
     Bin Node = RECORD
              Value : INTEGER;
              Left, Right : Bin Ptr;
     CB Ptr = ^CB Node;
     { Record structure for clustured binary tree }
     CB Node = RECORD
               Value : INTEGER;
              HLeft, HRight,
              LLeft, LRight : CB Ptr;
     REGTYPE = RECORD
                 AX, BX, CX, DX, BP,
                  DI, SI, DS, ED, FLAGS : INTEGER
                END;
     Time Rec = RECORD
                 HOUR, MIN, SEC, HSEC : BYTE
VAR Numbers : ARRAY [1..SIZE] OF INTEGER;
    Iter, I, Choice, CDV, Diff : INTEGER;
    BinTreeRoot : Bin Ptr;
    CBTreeRoot : CB_Ptr;
    Timer Start, Timer Stop, Time Elapsed: Time Rec;
PROCEDURE Get_Time(VAR TIME : Time Rec { output });
VAR REGISTER : REGTYPE;
    AH
              : BYTE:
BEGIN
  AH := $2C:
     WITH REGISTER, TIME DO BEGIN
        AX:= AH SHL 8;
        MSDOS (REGISTER);
        HOUR := Hi(CX);
        MIN := Lo(CX);
SEC := Hi(DX);
        HSEC := Lo (DX);
      END;
PROCEDURE Time_Diff(T_Start,
                    T Finish
                                   : Time Rec;
                                                 { input }
                    VAR Delta Time : Time Rec
```

```
BEGIN
 WITH Delta Time DO BEGIN
    HOUR := T Finish.HOUR - T Start.HOUR;
    IF T Start.MIN > T Finish.MIN THEN BEGIN
        HOUR := HOUR - 1;
        T Finish.MIN := T Finish.MIN + 60;
   END:
   MIN := T Finish.MIN - T Start.MIN;
    IF T Start.SEC > T Finish.SEC THEN BEGIN
        MIN := MIN - 1;
        T Finish.SEC := T Finish.SEC + 60;
    END:
    SEC := T Finish.SEC - T Start.SEC;
    IF T Start. HSEC > T Finish. HSEC THEN BEGIN
        SEC := SEC - 1;
        T Finish. HSEC := T Finish. HSEC + 100;
    END:
    HSEC := T Finish. HSEC - T Start. HSEC;
 END; (* WITH *)
END; (* Time Diff *)
PROCEDURE Show Time (T : Time Rec { input });
BEGIN
    WITH T DO BEGIN
        WRITE('Time elapsed = ',HOUR,' : ',MIN,' : ',SEC,'.',HSEC);
    END:
END; (* Show Time *)
PROCEDURE Create (Choice : INTEGER { input });
VAR J : INTEGER:
    Angle, FloatSize, Increment : REAL;
    CASE Choice OF
     1 : BEGIN
            Angle := 0.0;
             Increment := Pi / 360.0;
             FOR J := 1 TO SIZE DO BEGIN
                 Numbers[J] := Trunc(SIN(Angle) * RANGE);
                 Angle := Angle + Increment;
             END;
         END:
      2 : BEGIN
             Angle := 0.0;
             Increment := Pi / 360.0;
             FOR J := 1 TO SIZE DO BEGIN
                 Numbers[J] := Trunc(COS(Angle) * RANGE);
                 Angle := Angle + Increment;
             END;
         END:
      ELSE BEGIN
              Numbers[1] := RANGE div 2;
              FOR J := 2 TO SIZE DO
                  Numbers[J] := Trunc(Random * RANGE);
           END:
     END; { CASE }
  END;
                                               (continued on next page)
```

A Reconfigurable Programmer's Editor With Source Code

ME is a high quality programmer's text editor written specifically for the IBM PC. It contains features only found in the more expensive programmer's text editors. These features include:

- Multiple Windows
- Column cut and paste
- Line marking for source code
 - Regular Expressions
 - C-like Macro Language
 - Reconfigurable Keyboard
- Capture your DOS session
- Run your compiler and examine errors
- Comes with useful precompiled macros
 - UNIX-like CTAGS
 - 43-line mode with EGA

This is the ONLY editor with the power of the higher priced editors which comes with complete source code!

New commands and features may be added to the editor by writing programs in its macro language. The language resembles C, and comes with a rich set of primitives for handling strings and changing the editor environment, plus most of the flow-of-control constructs that come in C (for, while, if, break, continue).

The code is written in standard C, with several key library routines written in assembler for speed. The source code option is perfect for OEMs and VARs who want to add editing or word processing capabilities to their applications.

Price for editor and on-line documentation --- \$39.95

Price for editor with complete source code -- \$94.95

(Please add \$2 for shipping & handling, \$6 overseas)

Special offer -- New York Word word processor -- \$39.95 -- Multi-windowing, mail merge, hyphenation, math, regular expressions, TOC and index generators

MAGMA SYSTEMS

138-23 Hoover Ave. Jamaica, NY 11435 (201) 792 - 3954

CIRCLE 313 ON READER SERVICE CARD

STRUCTURED PROGRAMMING

Listing One (Listing continued, text begins on page 122.)

```
PROCEDURE Bin_Insert(VAR Root : Bin_Ptr; { input }
                          Item : INTEGER { input });
(* Insert element in binary-tree *)
BEGIN
    IF Root = NIL THEN BEGIN
        NEW (Root);
        Root^.Value := Item;
        Root^.Left := NIL;
        Root^.Right := NIL
    END
   ELSE
        WITH Root ^ DO
            IF Item < Value THEN Bin_Insert(Left, Item)</pre>
                             ELSE Bin Insert (Right, Item);
PROCEDURE Bin_Search(VAR Root : Bin_Ptr; { input }
                        Target : INTEGER { input });
(* Recursive procedure to search for Target value *)
BEGIN
    IF Root <> NIL THEN
        IF Target <> Root^.Value THEN
            IF Target < Root^.Value THEN BEGIN
   Root := Root^.Left;</pre>
                Bin_Search (Root, Target)
            END
            ELSE BEGIN
                Root := Root^.Right;
                 Bin Search (Root, Target)
END:
PROCEDURE CB_Insert (VAR Root : CB Ptr; { input }
                    VAR Item : INTEGER { input });
(* Insert element in a CB-tree *)
    IF Root = NIL THEN BEGIN
        NEW (Root);
        Root^.Value := Item;
        Root^.LLeft := NIL;
        Root^.LRight := NIL;
        Root^.HLeft := NIL;
        Root^.HRight := NIL;
    END
    ELSE
        WITH Root ^ DO BEGIN
            Diff := Value - Item;
            IF Diff > 0 THEN
                 IF ABS (Diff) <= CDV THEN CB Insert (LLeft, Item)
                                     ELSE CB_Insert (HLeft, Item)
                 IF ABS (Diff) <= CDV THEN CB Insert (LRight, Item)
                                     ELSE CB_Insert (HRight, Item);
        END; (* WITH *)
END;
PROCEDURE CB_Search(VAR Root : CB_Ptr; { input }
                     VAR Target : INTEGER { input });
(* Recursive procedure to search for Target value *)
BEGIN
```

```
IF Root <> NIL THEN
        IF Target <> Root^.Value THEN BEGIN
            Diff := Root^.Value - Target;
            IF Target < Root^. Value THEN BEGIN
                 IF ABS(Diff) <= CDV THEN Root := Root^.LLeft
                                      ELSE Root := Root^.HLeft;
                 CB Search (Root, Target)
            END
            ELSE BEGIN
                 IF ABS (Diff) <= CDV THEN Root := Root^.LRight
                                      ELSE Root := Root^.HRight;
                 CB Search (Root, Target)
            END;
        END:
END:
BEGIN (* MAIN *)
 CDV := Trunc (0.05 * RANGE);
 ClrScr:
 WRITE(' ':10);
 WRITELN('----- Menu for Method of Generating Numbers -----);
 WRITEIN:
 WRITELN(' 0) Random numbers ');
 WRITELN(' 1) Sine function ');
 WRITELN(' 2) Cosine function ');
 WRITELN:
 WRITE ('Enter code for array creation : ');
 READLN (Choice); WRITELN;
 Create (Choice);
 WRITELN; WRITELN('Created array'); WRITELN;
  (* Building the binary tree *)
 BinTreeRoot := NIL;
 Get Time (Timer Start);
 FOR I := 1 TO SIZE DO
    Bin Insert (BinTreeRoot, Numbers [I]);
 Get Time (Timer Stop);
 Time Diff(Timer_Start, Timer_Stop, Time_Elapsed);
 Show_Time (Time_Elapsed);
 WRITELN(' for creating binary Tree'); WRITELN;
  (* Building the CB-tree *)
 CBTreeRoot := NIL;
 Get Time (Timer Start);
 FOR I := 1 TO SIZE DO
    CB Insert (CBTreeRoot, Numbers[I]);
 Get Time (Timer Stop);
 Time Diff(Timer_Start, Timer_Stop, Time_Elapsed);
  Show Time (Time Elapsed);
 WRITELN(' for creating CB-Tree'); WRITELN;
 Get_Time(Timer_Start);
 FOR Iter := 1 TO MainLoopCount DO
      FOR I := SIZE DOWNTO 1 DO
          Bin Search (BinTreeRoot, Numbers [SIZE + 1 - I]);
 Get Time (Timer Stop);
 Time_Diff(Timer_Start, Timer_Stop, Time_Elapsed);
Show_Time(Time_Elapsed); WRITELN(' for binary tree search');
  WRITELN;
 Get Time(Timer_Start);
 FOR Iter := 1 TO MainLoopCount DO
         FOR I := SIZE DOWNTO 1 DO
          CB Search (CBTreeRoot, Numbers [SIZE + 1 - I]);
  Get Time (Timer Stop);
  Time_Diff(Timer_Start, Timer_Stop, Time_Elapsed);
  Show Time (Time Elapsed); WRITELN(' for CB-tree search');
  WRITELN;
  WRITELN ('DONE'); WRITELN;
                                                         End Listing One
FND -
```

(Listing Two begins on next page.)

TRUE MULTITASKING

With

MultiDos Plus

"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/ resume programs.
- Programmatic interface via INT 15H for the following.
 - Intertask message communication. Send/receive/check message present on 64 message queues.
 - Task control by means of semaphores. Get/release/check semaphores.
 - Change priority-256 priority levels.
 - Suspend task for specified interval.
 - Spawn and terminate external and internal tasks.
 - Disable/enable multitasking.
 - and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: \$24.95 OR \$99.95 with source code

Outside USA add \$5.00 shipping and handling. Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT

13 Westfield Rd, Natick, MA 01760

MA orders add 5% sales tax.

CIRCLE 309 ON READER SERVICE CARD

68020

UNIX COMPATIBLE **MULTIUSER SYSTEM**

The MPULSE Model 20

Multi-Processor Design:

Separating application processing from I/O is a well understood goal in multiuser supermicrocomputer design. As the real UNIX system bottleneck is disk I/O bandwidth, not raw processor speed, LPC has spent a great deal of time and development effort in optimizing disk I/O throughput. The result is a disk I/O subsystem unparalleled in the under \$20,000 microcomputer class.

A fully asynchronous, high speed DMA channel links the MC68020 to the dual MC68000 I/O processors. A full 2 Mbytes of I/O processor memory is available for disk caching. The disk cache utilizes a least recently used, delayed write algorithm to achieve hit rates exceding 90%.

In addition to disk caching, LPC has extended the conventional UNIX caching mechanism with a new virtual caching technique, implemented in the kernel itself. The Dynamic Kernel Cache (DKC) distributes available memory between user processes and the internal UNIX cache. This dynamic allocation technique allows a much more efficient use of main memory with cache efficiency increased to over 80%, much higher than the conventional UNIX caching mechanism.

Operating System:

The MPULSE Model 20 hosts LPC's System V operating system, derived from the industry standard UNIX System V. The MPULSE System V port is tailored to complement the MPULSE hardware while retaining compatibility at all levels with the generic UNIX System V operating system. Areas of optimization and additional features include:

- Full demand-paged virtual memory
- Kernel portions of the terminal and mass storage I/O disciplines are distributed to the appropriate I/O processors
- Berkeley enhancements
- Dynamically distributed ramdisks
 On-line Winchester disk bad block replacement
- On-line device configuration
- True multisector I/O for streaming tape operation
- Support for implementing concurrent operating systems
- General purpose user-accessible SCSI device driver Automatic bi-directional modem support
- MWindows windowing capability

Base System Includes:

- 16 MHz MC68020 host processor
- DUAL 12 MHz MC68000 I/O sub-system
- 4Mbytes main memory
 2 Mbytes of dedicated disk cache memory
- Demand-Paged Virtual Memory
- Sophisticated LRU caching algorithm
- Dynamic Kernel Cache (DKC)
- MWindows 50 MByte hard disk expandable to 0.5 gigabytes
- 800 Kbyte floppy drive
- 60 Mbyte cassette tape backup
 8 serial ports expandable to 40 serial ports
- LPC System V derived from UNIX System V
- Berkeley Enhancements
- NCR Tower object code compatibility

Base System Price:

\$5995.00

Call (214) 340-5172

Warranty:

90 day (extended warranty available) 15 Day money back evaluation period

LPC Logic Process Corporation 10355 Brockwood Road Dallas, Texas 75238

DKC and MWindows are trademarks of LPC. UNIX is a trademark of AT&T Tower is a trademark of NCR.

CIRCLE 169 ON READER SERVICE CARD

STRUCTURED PROGRAMMING

```
Listing Two (Text begins on page 122.)
```

```
PROGRAM Test Clustered Lists;
{$R+, V-}
  Turbo Pascal program that implements routine for inserting,
  searching and viewing clustered list structures.
  Copyright (c) 1987 Namir Clement Shammas
CONST LENSTRING = 30;
     MAX LIST = 100;
 MDNMØ
      CDV = 0; { Critical Difference value }
TYPE LSTRING = STRING[LENSTRING];
    KeyArray = ARRAY [1..MAX_LIST] OF LSTRING;
     ListPtr = ^ListNode;
     { CLustered List structure }
     ListNode = RECORD
                 Key : LSTRING;
                  { other fields may be placed here }
                 NextPtr, NextHi : ListPtr;
VAR Head : ListPtr;
    LesKeys : KeyArray;
    I, Count : INTEGER;
{-----}
PROCEDURE Search Node (Head
                                 : ListPtr; { input }
                     SearchData : LSTRING; { input }
                 VAR Found
                                 : BOOLEAN; { output }
                 VAR LastPtr.
                     ThisPtr
                                : ListPtr { output });
{ search for 'SearchData' in list }
VAR HiTrack : BOOLEAN;
   Ordl, Diff : INTEGER;
BEGIN
    Found := FALSE;
    HiTrack := TRUE;
    LastPtr := NIL;
   ThisPtr := Head;
    Ord1 := ORD (SearchData[1]);
    WHILE (ThisPtr <> NIL) AND (ThisPtr^.Key < SearchData) DO BEGIN
       LastPtr := ThisPtr;
        TF HiTrack THEN BEGIN
            Diff := ORD (ThisPtr^.Key[1]) - Ord1;
            IF ABS (Diff) > CDV
               ThisPtr := ThisPtr^.NextHi
           ELSE BEGIN
               ThisPtr := ThisPtr^.NextPtr;
               HiTrack := FALSE { switch to low track }
```

```
END; { IF ABS(Diff) }
       END
       FLSE
           ThisPtr := ThisPtr^.NextPtr;
       { END IF HiTrack }
    END; { WHILE }
    IF ThisPtr <> NIL THEN Found := (ThisPtr^.Key = SearchData);
END; { Search Node }
PROCEDURE Insert List(VAR Head : ListPtr; { in/out }
                       NewData : LSTRING { input });
{ Insert new data string into the list }
VAR Found : BOOLEAN:
    Ordl, Diff : INTEGER;
    Tempo : LSTRING;
    Node, LastPtr, ThisPtr : ListPtr;
    Ord1 := ORD(NewData[1]); { get ascii code of firt char }
    IF Head = NIL THEN BEGIN { start a new list }
        new (Head):
        WITH Head DO BEGIN
            NextPtr := NIL;
            NextHi := NIL;
            Key := NewData
        END; { WITH }
    FND
    ELSE BEGIN { expand list }
        new (Node);
        WITH Node DO BEGIN
            Key := NewData;
            NextPtr := NIL;
            NextHi := NIL
         END; { WITH }
         Search_Node (Head, Node^.Key, Found, LastPtr, ThisPtr);
         IF LastPtr = NIL THEN BEGIN { insert as new list head }
             Diff := ORD (Head^.Key[1]) - Ord1;
             IF ABS (DIFF) > CDV THEN
                 Node^.NextHi := Head
                 Node^.NextHi := Head^.NextHi;
                 Node^.NextPtr := Head;
             { END IF }
             Head := Node;
         END
         ELSE BEGIN { insert new data in the middle or at the tail }
             Diff := Ord1 - ORD (LastPtr^.Key[1]);
             IF Diff <= CDV THEN BEGIN
                  { insert inside a clustered sublist }
                  { LasPtr may be a high of low track node }
                 Node^.NextPtr := LastPtr^.NextPtr;
                 LastPtr^.NextPtr := Node
             ELSE BEGIN
```

(continued on next page)



SQL Compatible Query System adaptable to any operating environment.

CQL Query System. A subset of the Structured English Query Language (SEQUEL, or SQL) developed by IBM. Linked files, stored views, and nested queries result in a complete query capability. File system interaction isolated in an interface module. Extensive documentation guides user development of interfaces to other record oriented file handlers.

Portable Application Support System

Portable Windowing System. Hardware independent windowing system with borders, attributes, horizontal and vertical scrolling. User can construct interface file for any hardware. Interfaces provided for PC/XT/AT (screen memory interface and BIOS only interface), MS-DOS generic (using ANSI.SYS), Xenix (both with and without using the curses interface), and C-library (no attributes).

Screen I/O, Report, and Form Generation Systems. Field level interface between application programs, the Query System, and the file system. Complete input/output formatting and control, automatic scrolling on screens and automatic pagination on forms, process intervention points. Seven field types: 8-bit unsigned binary, 16 bit signed binary, 16 bit unsigned binary, 32 bit signed binary, monetary (based on 32 bit binary), string, and date.

Including Source Code

\$395.00

File System interfaces include C-tree and BTRIEVE.

HARDWARE AND FILE SYSTEM INDEPENDENT

MACHINE INDEPENDENT SOFTWARE CORPORATION

1415 NORTHGATE SQUARE #21D RESTON, VA 22090

VISA/Master Charge accepted (703) 435-0413

*C-tree is a trademark of FairCom

IBM, SEQUEL, PC, XT, AT are trademarks of IBM Corp. MS-DOS and Xenix are trademarks of Microsoft Corp.

CQL and the CQL logo are trademarks of Kurtzberg Computer Systems.

CIRCLE 294 ON READER SERVICE CARD

STRUCTURED PROGRAMMING

Listing Two (Listing continued, text begins on page 122.) IF ThisPtr <> NIL THEN BEGIN Diff := Ord1 - ORD(ThisPtr^.Key[1]); IF ABS (Diff) > CDV THEN BEGIN {insert between two high track nodes } Node^.NextHi := LastPtr^.NextHi; LastPtr^.NextHi := Node END ELSE BEGIN (swap names in the next high track node) Tempo := Node^.Key; Node . Key := ThisPtr . Key; ThisPtr^.Key := Tempo; { insert a new swapped first element } { in clustered sublist Node^.NextPtr := ThisPtr^.NextPtr; ThisPtr^.NextPtr := Node END; { IF } END ELSE BEGIN { insert as last high track node } Node^.NextHi := LastPtr^.NextHi; LastPtr^.NextHi := Node END; { IF } END; { IF } END; { IF LastPtr = NIL } END; { IF Head = NIL } END; { Insert List } {-----} (Head : ListPtr; { input }
VAR Keys : KeyArray; { output } PROCEDURE List to Array (Head VAR Count : INTEGER { output }); { Converts the list to an array containing sorted names } {-----} PROCEDURE Visit_Low_Node (VAR Node : ListPtr); { Local recursive routine to visit low tracks of a list } BEGIN IF (Node <> NIL) AND (Count < MAX_LIST) THEN BEGIN Count := Count + 1; Keys[Count] := Node^.Key; WRITE(' ', Keys[Count]:10); Visit_Low_Node(Node^.NextPtr); END ELSE WRITELN(' -]'); END; { Visit Low Node } PROCEDURE Visit_Hi_Node (VAR Node : ListPtr); { Local recursive routine to visit high tracks of a list } BEGIN IF (Node <> NIL) AND (Count < MAX_LIST) THEN BEGIN Count := Count + 1; Keys[Count] := Node^.Key; WRITE (Keys [Count]:10); Visit_Low_Node(Node^.NextPtr); Visit Hi Node (Node^.NextHi); END:

```
END; { Visit Hi Node }
REGIN
    IF Head <> NIL THEN BEGIN
        Count := 0;
        Visit Hi Node (Head);
    END
    ELSE
        Count := 0;
    { END IF }
END; { List to Array }
BEGIN
    ClrScr;
    IF CDV < 0 THEN BEGIN
         WRITELN('Adjust Critical Difference Value Please');
         HALT
     END;
     Head := NIL;
     WRITELN('List of sorted capitals '); WRITELN;
     Insert List(Head, 'Sau Paulo'); Insert List(Head, 'Moscow');
Insert List(Head, 'Otawa'); Insert List(Head, 'Tokyo');
Insert List(Head, 'Bern'); Insert List(Head, 'Warsaw');
     Insert_List(Head, 'Bern');
     Insert List (Head, 'Madrid'); Insert List (Head, 'Lisbon');
Insert List (Head, 'Paris'); Insert List (Head, 'Baghdad');
     Insert List(Head, 'Paris');
     List to Array (Head, LesKeys, Count);
END.
```

End Listings

Periscope Power... Made to Order!

The unquestioned price/performance leader in PC debuggers offers you a full line of debugging systems, from software-only Periscope II-X to the full-fledged hardware breakpoint Periscope III.

Periscope debugging power is available in four models... Start with the model that fits your current needs. When you need more horsepower, upgrade for the difference in price plus \$10!

When you move to another Periscope model, don't worry about having a lot more to learn... Even when you move to the most powerful model, Periscope III, an extra dozen commands are all that's involved.

A Periscope I user who recently began using Periscope III writes, "I like the fact that within the first half hour of use I was debugging my program instead of learning to use the debugger."

Periscope's software is solid, comprehensive, and flexible. It helps you debug just about any kind of program you can write...thoroughly and efficiently.

Periscope requires an IBM PC, XT, AT, or close compatible (Periscope III requires hardware as well as software compatibility); DOS 2.0 or later; 64K available memory; one disk drive; an 80-column monitor.

Call us with your questions; we'll be happy to send you FREE information or help you decide on the model that best fits your needs. **Periscope I** includes a half-length board with 56K of write-protected RAM; break-out switch; software and manual for \$345.

 $\textbf{Periscope II} \ \, \text{includes break-out switch; software and manual for \$175.}$

Periscope II-X includes software and manual (no hardware) for \$145.

Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 8 MHz is \$995; for machines running up to 10 MHz, \$1095.

Order Your Periscope, Toll-Free, Today!



14 Bonnie Lane • Atlanta, GA 30328 • 404/256-3860

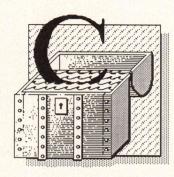
The Ultimate Metronome: Writing Interrupt Service Routines in C

he ostensible topic of this month's column is a fancy metronome program capable of doing polyrhythms and even more complex rhythmic patterns. As is often the case, however, the pieces are as interesting as the whole. For example, in order to get adequate timing resolution, I had to speed up the IBM PC's system clock and then supply my own interrupt service routine, written largely in C-though there's a little assembly language. The routines used for this purpose are useful in other applications as well (such as a preemptive multitasking scheduler that I'm also writing).

The metronome is pretty spiffy in its own right, though. I use it for my own compositional work to create "click tracks" for electronic pieces. (A click track is one track of a multitrack tape that holds nothing but timing information. You use it as a reference as you add additional tracks to the piece and then remove it from the final mix.) The program lets you plot all the complex rhythmic patterns required for a complete piece of music. (The maximum time varies; the program can do up to 1,280 measures, and if these are measures of 6/

by Allen Holub

8 at metronome 120, that's 64 minutes.) Though I'm just ringing the bell on the PC to mark beats, it wouldn't



be too hard to modify this program to use a MIDI drum machine or the like to make the sounds.

Metronomes and Time Signatures

When metronomes were introduced in the early 19th century (Beethoven was the first major composer to use one), it finally became possible for composers to present timing information accurately to performers. To understand why this wasn't possible before, you have to look at how a standard time signature works. Music, by nature, is cyclical, and a measure is one complete cycle of the underlying rhythmic pattern. A march, for example, has two beats in a measure (oom pah, oom pah), a waltz has three (oom pah pah, oom paa pah), and rock and roll usually has fourbeat measures. The main purpose of the measure structure is to tell the performer where to put emphasis. That is, the first note of the measure (called the downbeat) is usually played a little more forcefully than the other notes in the measure (OOM! pah pah).

Coupled with the measure are the individual notes. A whole note requires an entire measure to play. If you think in terms of an organ, you hold the key down for the entire measure. A half note requires half a measure. There are two half notes in a whole note, two quarter notes in a half note, two eighth notes in a quarter note, and so forth. Each of these types of notes are represented by a slightly different symbol in a musical score. You'll note (so to speak) that this system is self-relative. There's no absolute durational information in a particular note symbol-everything's relative to the other notes.

A standard time signature gives you two pieces of information—the number of beats in a measure and which of the various types of notes in the score represent one beat. For example, the time signature 3/4 (pronounced "three four") has three beats in the measure and a quarter note is one beat long. So a complete three-beat measure can be composed of three quarter notes, six eighth notes, two quarter notes and two eighth notes, and so forth. A time signature is not a fraction; 6/8 is not the same thing as 3/4. The emphasis in the former is on every sixth beat, whereas the emphasis in the latter is on every third beat. There's no information in a time signature about the actual duration of a beat in seconds. This duration is usually detailed with written instructions-most often in Italian-that are quite inexact (fast, slow, medium-fast, and so on).

The metronome changed all this. The mechanical metronome used by Beethoven is still in use today. It consists of a spring-driven pendulum with a weight on it. The position of the weight determines the rate at

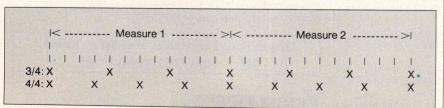


Figure 1: A 3 against 4 polyrhythm

Announcing - the database development system that you designed.



PROGRAMMERS-

We asked what you wanted in a database development system and we built it!

db_VISTA III™ is the database development system for programmers who want powerful, high performance DBMS capabilities ... and in any environment. Based on the network database model and the B-tree indexing method, db_VISTA III gives you the most powerful and efficient system for data organization and access. From simple file management to complex database structures with millions of records. db_VISTA III runs on most computers and operating systems like MS-DOS, UNIX, VAX/VMS and OS/2. It's written in C and the complete source code is available, so your application performance and portability are guaranteed! With db_VISTA III you can build applications for single-user microcomputers to multi-user LANs, up to minis and even mainframes.

The db_VISTA III™ Database Development System

db_VISTA": The High Performance DBMS

The major features include:

- computers.
- Multiple database access.
- · File and record locking.
- Automatic database recovery
- Transaction processing and logging.
- Timestamping.
- Database consistency check utility.
- Fast access methods based on the network database model and B-tree indexing. Uses both direct "set" relations and B-tree indexing independently for design flexibility and performance.
- An easy-to-use interactive database access
- File transfer utilities for importing/exporting ASCII text and dBASE II/III files
- A Database Definition Language patterned after C
- Virtual memory disk caching for fast

- A runtime library of over 100 functions
- Multi-user support for LANs and multi-user
 Operating systems: MS-DOS, UNIX V, XENIX, VMS, OS/2.
 - · C Compilers: Lattice, Microsoft, IBM, Aztec, Computer Innovations, Turbo C, XENIX, and UNIX
 - LAN systems: LifeNet, NetWare, PC Network, 3Com, SCO XENIX-NET, other NET-BIOS compatible MS-DOS networks.

2 db_QUERY:™ The SQL-based Query.

- Provides relational view of db_VISTA applications.
- Structured Query Language
- C linkable.
- Predefine query procedures or run ad-hoc queries "on the fly

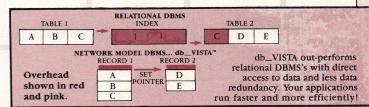
3 db_REVISE™: The Database Restructure Program.

- Redesign your database easily.
- · Converts all existing data to revised design.

All components feature royalty-free run-time distribution, source code availability and our commitment to customer service. That's why corporations like ARCO, AT&T, Hewlett-Packard, IBM, Northwestern Mutual Life, UNISYS and others use our products.

RAIMA'S COMMITMENT TO YOU: No Royalties, Source Code Availability, 60 days FREE Technical Support and our 30-day Money-Back Guarantee. Extended services available include: Application Development, Product Development, Professional Consulting, Training Classes and Extended Application Development Support.

HOW TO ORDER: Purchase only those components you need. Start out with Single-user for MS-DOS then add components, upgrade ... or purchase Multi-user with Source for the entire db_VISTA III System. It's easy... call toll-free today!



db_VISTA III™ Database **Development System**

db_VISTA III \$595 - 3960 \$595 - 3960 db_QUERY db_REVISE" \$595 - 3960 db_VISTA™ File Manager Starts at \$195

We'll answer your questions, help determine your needs and get you started.



CALL TODAY! 1-800-db-RAIMA



(that's 1-800-327-2462)



3055 112th Avenue N.E., Bellevue, WA 98004 (206)828-4636 Telex: 6503018237MCIUW FAX: (206)828-3131

A Different BASIC Might Make All the Difference

We'll skip the four-color gatefold. And the extravagant claims. Because if you're serious about programming, you just want the straight facts:

	True BASIC 2.01	Microsoft Quick Basic 3.0	Borland Turbo Basic 2.0
GRAPHICS			
Supports Hercules Graphics	YES	NO	NO
Device-Independent Graphics Syntax	YES	NO	NO
User-Defined Coordinates	YES	LIMITED	LIMITED
Matrix Graphics Coordinates	YES	NO	NO
ARRAY HANDLING			Mary Const
Matrix Algebra	YES	NO	NO
Maximum Numeric Array	UNLIMITED	64K	64K
Max. Number of Array Dimensions	255	63	8
Max. Number of Elements/Dimension	UNLIMITED	32K	32K
Dynamic Redimensioning	YES	NO	NO
Matrix I/O Statements	YES	NO	NO
STRING/FILE HANDLING		ATTEM N	
Maximum String Length	64K	32K	32K
Total String Space	UNLIMITED	64K	64K
Maximum Record Size	16MB	32K	32K
Max. Bytes/Binary File Read	64K	NA	32K
PRODUCTIVITY ENHANCERS		TO PROPERTY.	
Modules	YES	NO	NO
Separately Compiled Libraries	YES	LIMITED	NO
Workspaces	YES	NO	NO
Immediate Mode	YES	NO	•NO
SPECIAL FEATURES			Mark State
Stop/Continue Execution	YES	NO	NO
Max. Source File	UNLIMITED	UNLIMITED	64K
Script Files	YES	NO	NO
Keystroke Macros	YES	NO	NO
Max. Characters/Line	64K	255 char.	249 char.
Max. Scalar Data Space	UNLIMITED	64K	64K
Mouse Support	YES	NO .	NO
80386 Version	YES	NO	NO
Portability to:	Macintosh, Amiga, Atari	Translation required	No other machines

Three very structured, very powerful programming packages. All with fancy editors and fast compilers. Two of them are the same in other respects. And one of them, True BASIC, is quite a bit different. With syntax and features that will make you more productive.

That's why reviewers for magazines like BYTE, PC Tech Journal and Computerworld keep giving True BASIC their top marks. And why OEMs pick True BASIC after they've evaluated all the others. See why True BASIC can make the difference for you.

Call 1-800-TRBASIC today.

True BASIC, Quick Basic and Turbo Basic are trademarks of True BASIC, Inc, Microsoft and Borland, respectively. Macintosh, Amiga and Atari are trademarks of Apple Computer, Inc., Commodore-Amiga, Inc. and Atari Corporation. Copyright 1987 True BASIC. Specifications are accurate as of August 1987.



39 SOUTH MAIN STREET HANOVER, N.H. 03755 (603) 643-3882

CIRCLE 344 ON READER SERVICE CARD

C CHEST

(continued from page 106)

which the pendulum swings, and the box ticks at the end of each swing. Metronomes are calibrated in beats per minute, so a metronome 60 is one beat per second and a metronome 120 is two beats per second. The composer can now put an instruction at the top of the score saying something such as "an eighth note is played at metronome 100."

Traditional metronomes are fine if you're writing waltzes, but they aren't really adequate for most modern music for three reasons. First, some pieces have different time signatures on literally every measure, and you obviously can't stop after every measure to adjust your metronome. A second, related, problem is a piece with measures of alternating time signatures—say, alternating measures of 5/8 and 6/8. Dave Brubeck and Steve Reich both use this technique extensively. Finally, there's the problem of polyrhythms. A polyrhythm is a rhythmic pattern in which two dissimilar time signatures are superimposed. For example, in three against four, a three-beat pattern is played in one hand and a four-beat pattern in the other. These come together in at least one place (usually on the downbeat) of every measure. This timing is illustrated in Figure 1, page 106. The three-beat pattern beats on every fourth tick, the four-beat pattern on every third tick. Polyrhythmic music is usually great dance music-you hear it in Latin, Brazilian, and West African pop music and more and more in contemporary Western music.

Click: A Four-Channel Programmable Metronome

The program I present this month, called click, is a programmable metronome that solves all the problems I've just discussed. Its invocation syntax is:

click [-d] file

where file holds an input program. Click outputs two things-beats in the form of sounds and a running count of the number of measures played so far.

Click is programmed using a multi-

track tape recorder model. Four tracks (numbered 0 to 3) are supported, each of which can contain up to 1.280 measures. Track 0 is always required; the others are optional. The four tracks can be programmed independently, or they can synchronize to each other at the beginning of each measure. Different notes are used for the different tracks so that you can distinguish them. Track 0 uses middle C, track 1 uses the C one octave up, track 2 uses the G above that, and track 3 uses the C above that. When there's a conflict (a beat occurring at the same time on more than one track), the note associated with the higher track is used. The optional -d command-line switch forces all four tracks to use the same note (C5) for their beats.

A short click program is shown in Example 1, page 112. The input language works as follows. White space (spaces, tabs, blank lines, and so on) is ignored except as is needed to separate commands. By the same token, all lines that don't start with the word *track* and all text on a line that follows a semicolon are ignored. You can use this mechanism to comment your programs. The virtual tape is assembled using *track* commands, which take the following form:

track n: <measure>[, <measure>]

where *n* is a number in the range 0–3 that specifies a track on the virtual tape. A <*measure*> is all text starting with a colon or comma, up to the next comma or end of line. Several comma-delimited measures can be placed on a single line. Tracks are accumulated by successively adding the information in each *track* command to the indicated track.

A measure comprises one or more of the following commands, where *n* and *m* are numbers, and the commands can occur in any order:

n/m—A command that starts with a digit represents either the time signature or the number of beats in the measure. Here, n is the number of beats in the current measure and /m is ignored. In fact, you can omit the /m entirely if you like. A time signature is required in every measure—I'll show you some examples in a mo-

NEW VERSION

PC/VI

UNIX's VI Editor Now Available For Your PC!

Are you being as productive as you can be with your computer? An editor should be a tool, not an obstacle to getting the job done. Increase your productivity today by choosing **PC/VI**—a COMPLETE implementation of UNIX* VI version 3.9 (as provided with System V Release 2).

PC/VI is an implementation of the most powerful and most widely used full-screen editor available under the UNIX operating system. The following is only a hint of the power behind **PC/VI**:

- · Global search or search and replace using regular expressions
- · Full undo capability
- Deletions, changes and cursor positioning on character, word, line, sentence, paragraph, section or global basis
- · Editing of files larger than available memory
- Shell escapes to DOS

CHMOD • DIFFH

- Copying and moving text
- Macros and Word abbreviations
- Many controllable options including Auto-indent, Showmatch, and Wrap Margin
- Filter text through external programs AND MORE!

Don't take it from us. Here's what some of our customers say: "Just what I was looking for!", "It's great!", "Just like the real VI!". "The documentation is so good I have already learned things about VI that I never knew before." — *IEEE Software*, September 1986.

PC/VI is available for IBM-PC's and generic MS-DOS[†] systems for only \$149. Included are CTAGS and SPLIT utilities, TERMCAP function library, and an IBM-PC specific version which enhances performance by as much as TEN FOLD!

PC/TOOLS"

What makes UNIX so powerful? Sleek, Fast, and **POWERFUL** utilities! UNIX gives the user not dozens, but hundreds of tools. Now the most powerful and popular of these are available for your PC! Each is a complete implementation of the UNIX program. Open up our toolbox and find:

ASA COMM DIFF3 MV SEE TR
BANNER CP FIND OD SORT TOUCH
BFS CUT GREP PASTE SPLIT UNIQ
CAL DATE HEAD PR STRINGS WC

· SED

TAIL

All of these for a limited introductory price of only \$49.00; naturally, extensive documentation is included!

• MAKE

PC/SPELL

Why settle for a spelling checker which can only compare words against its limited dictionary database when **PC/SPELL** is now available? **PC/SPELL** is a complete implementation of the UNIX spelling checker, renowned for its understanding of the rules of English! **PC/SPELL** determines if a word is correctly spelled by not only checking its database, but also by testing such transformations as pluralization and the addition and deletion of prefixes and suffixes. For only \$49.00, **PC/SPELL** is the first and last spelling checker you will ever need!

Buy **PC/VI** and **PC/TOOLS** *now* and get **PC/SPELL** for only \$1.00! Site licenses are available. Dealer inquiries invited. MA residents add 5% sales tax. AMEX, MC and Visa accepted without surcharge. Thirty day money back guarantee if not satisfied! Available in 5½", 3½" and 8" disk formats. For more information call today! *UNIX is a trademark of AT&T. *MS-DOS is a trademark of Microsoft.

CUSTOM SOFTWARE SYSTEMS

P.O. BOX 678 • NATICK, MA 01760 617 • 653 • 2555



CIRCLE 268 ON READER SERVICE CARD

Breakthrough in interface management. Generate C code from Dan Bricklin's Demo screens. Date fields. Full color support. Money fields. Fully programmable field behavior. Scrolling text within fields. Calculator style numeric input. User definable entry validation. Field marking. Orthogonal field movement. Specify fields by number or location. Source code included. Screen sizes limited only by memory. Interfaces with db_VISTA and other libraries. Text style numeric input. Input masking. List fields. Create spreadsheets. Includes Look & Feel screen designer. Integer fields. String formatting commands. Date and time validation functions. Generate C code with Look & Feel screen designer. Supports automatic vertical and

horizontal scrolling. Clean screen fields per screen limited only by development. String fields. Easy to painting. Bind as much data as de data entry with commas. Ask a gramming library. Hexadecimal or No fields. Float fields. Quick C. Speaker functions. Lattice. Create Slug. Numeric validation routines. keystroke level. Customize screens 30 day money back guarantee. Gen assortment of editing commands. windows. Assign validation data to credentials. Pull down menus. Sup mode. All functions are kept in C style function reference. Pop-up functions. Numeric range checking. tive function names. Date and time Capture screens from existing as deep as desired. Easy to main checking. Date and time conver definition language based on C's ly definable borders. The current cally highlighted. Create reports.

C-scape 2.0 Cook & Feel

The state-of-the-art interface management system preferred by professional C programmers and consultants worldwide.

borders. Se

sor types.

EGA, and

Aztec. In

for writing to

cludes an

ver. A vari

functions.

Multi-level

Borders with

eo RAM dri

New device

ated. Color

ical use of

with prompt

grated help

many screens

data entry

follow man

Exploding borders. Convert old programs to C. Borders with titles. Color map enables use of logical colors. Toll-free telephone support line. 24 hour bulletin board. Automatically detects type of monitor being used. ANSI driver included. Screen and field definitions. Uses device drivers for portability. View text in pop-up windows. Read only fields. Rich assortment of editing commands. Pass-

word entry fields. paging functions Customizable lect different cur Supports CGA, monochrome. cludes functions the display. In ANSI device dri ety of keyboard Lined borders. menuing systems. scroll lights. Vid ver included. drivers can be cre map enables log colors. Borders lines. Fully inte system. Create as as needed. Create screens. Easy to ual. Professional

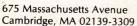
Rook & Feel

- WYSIWYG screen design tool
- Generates readable C code
- Create menus and data entry
- Define fields of any type
- Variables, prompts, and validation
- Line draw and erase
- Block, move, cut, paste, copy
- Horizontal and vertical scrolling
- Edit Dan Bricklin Demo slides
- Full color support
- Fast, easy, and fun to use
- Includes help
- Full-feature demo available

customer sup port. Includes higher level

functions. Device drivers swappable at run-time. Context sensitive help system. Cross referenced help screens. Protected fields. Object-oriented design. Read in screen defini tions from disk files. Digitally mastered. Assign prompt strings to fields. UNIX. No runtime license. Numeric range checking. Unified field theory. Full printf % substitution

Oakland Group, Inc.



800-233-3733 617-491-7311

CALL NOW



PC/MS-DOS \$279, plus shipping (includes C-scape, Look & Feel, source, manual and support). UNIX/others call. 30-day review.

C-scape 2.0

machine. Number of

memory. Fast screen

modify. Fast screen

sired to fields. Numeric bout our linear pro

fields. Long fields. Yes

Read only fields.

reports. Codename

Validate data at the

and menus at run time.

eric data pointer. Rich

Easy to learn. Pop-up

fields. Corporate C

ports EGA 43 line

separate modules. Full

prompt and message

No royalties. Descrip-

conversion routines.

programs. Nest screens

tain. Run time error

sion functions. Screen

printf. Time fields. Ful-

field can be automati-

Windows, windows, windows

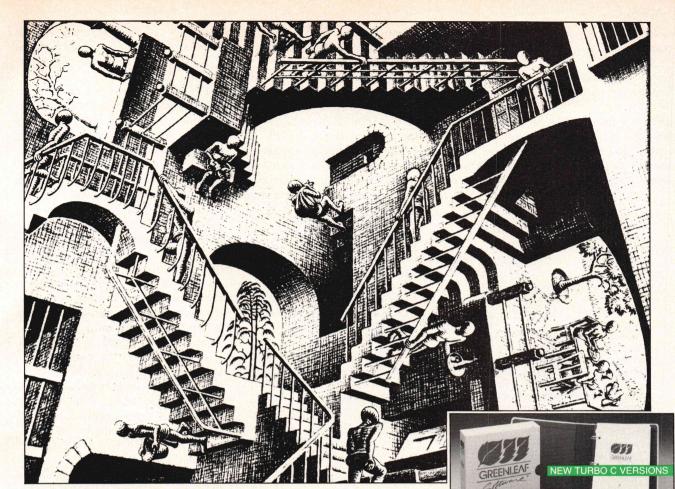
Includes ROM BIOS driver. Fields can support any data type. Scrolling/

included. Specify writeable and non-writeable positions within fields.

- Menus, menus, menus
- Vast help system
- Create any type of field
- Data entry and validation
- Smart borders
- Extensive function library
- Swappable device drivers
- Easy to learn and use
- Easy to maintain and modify
- Unsurpassed flexibility
- Professional manual
- No royalties; no run-time license
- Source code included
- Demo package available

within screen definitions. Supports all memory models. C Bricklin run. Turbo C. 24 hour bulletin board. Higher level functions included. Object-oriented design. All library functions are kept in separate modules. Nest screens as deep as desired. Design screens with Look & Feel screen designer. New device drivers can be created. Create as many screens as needed. No run-time license. Hexadecimal fields. Preferred by professionals and consultants. Microsoft. Cross referenced help system. by space aliens. Generate C code with Look & Feel screen designer. Context sensitive help system. Scroll lights. Read in screen definitions from disk files. Automatic vertical and horizontal scrolling. Batteries not included. Double and single line borders. Cross-referenced help system. Save and restore regions of the display. Nested menus. Quick C. Create screens from ASCII files. Easy to learn and use. Horizontal and vertical scrolling. Used by consultants and corporations worldwide. Easy to maintain. Professional documentation. Screen designer creates

readable C code. Portable. Easily modifiable functions. No royalties. Source code included. Turn Dan Bricklin slides into C. Professional support. Interface examples for data base management. Validation at keystroke level. Vast integrated and indexed context-sensitive help system. Save and restore regions of the display. Now supporting Quick C, Turbo C, Aztec, Lattice, Microsoft, UNIX and others. And that's not all. Call for demo.



Avoid extra steps.

You've got better things to do than repeat the same steps. Over . . . and over . . . and over. Up your productivity with Greenleaf Software.

With more than 70 new functions added to our popular libraries, Greenleaf is now the most complete and mature C language function resource available. It's no wonder we've been rated the best. Winning program developers in major corporations such as IBM, EDS and GM have proven our reliability in thousands of applications.

Step Lively

New Greenleaf Functions v.3.10 includes 295 of the functions you've been asking for — DOS, disk, video, color text and graphics, string, time/date, keyboard, plus many more! With Greenleaf, you'll finish faster.

Cut Corners

When it comes to merging information, the new Greenleaf Comm Library v.2.10 is the fastest communications facility of its kind. Over 120 functions — ring buffered,

interrupt-driven asynchronous communications. And, only Greenleaf gives you the power to build a 16-port communication system.

Get on the Fast Track

Order your new Greenleaf library today! See your dealer or call 1-800-523-9830.

Greenleaf Comm Library \$185.0	
Greenleaf Functions \$185.0	
Greenleaf DataWindows \$225.0	00
Greenleaf C Sampler \$ 94.5	50
Digiboard Comm4 \$325.0	00
Digiboard Comm8 \$535.0	00

In stock, shipped next day.

Greenleaf DataWindows and Turbo C

DataWindows, the finest C programming windows tool available, puts windows, transaction data entry and menus at your fingertips.

Our new TURBO C versions are ready to get you going fast! And, our new 3-in-1 C Sampler for only \$94.50 supports both Turbo C and Quick C with comm, windows, menus and more! Our libraries support all popular C compilers for MS DOS.

Call Toll Free:

800-523-9830

In Texas and Alaska:

GREENLEAF

214-446-8641

Greenleaf Software, Inc. 16475 Dallas Parkway, Suite 570 Dallas, Texas 75248

C CHEST

(continued from page 109)

ment. The maximum number of beats in a measure (n) is 255.

@n—A number following an at sign (@) is used to set the metronome count. For example, you'd use the following to play 100 beats at metronome 120:

track 0: 100 @120

One measure of 3/4, with the quarter note at metronome 120, is represented as follows:

track 0: 3/4 @120

Alternating measures of 5/8 and 6/8 at different metronome counts can be done with the following:

track 0: 6/8 @120, 5/8 @100 track 0: 6/8 @120, 5/8 @100 track 0: 6/8 @120, 5/8 @100

Click supports a resolution as low as one metronome count—a metro-

nome 1 is one beat/minute. There is no effective maximum count because the maximum is in the audio-frequency range.

The @ command is required for every measure on track 0. It's optional, however, on the other tracks. If the @ command is omitted, the time between beats is stretched out so that the measure takes up the same time as the corresponding measure on track 0. For example, you can take advantage of the higher note being used in a conflict to accent the downbeat of a measure as follows:

track 0: 3/4 @120, 3/4 @120 track 1: 1/4 , 1/4

Here, each measure of track 1 takes the same amount of time to play as the corresponding measure of track 0. You could do the same thing with:

track 0: 3/4 @120, 3/4 @120

track 1: 1/4 @ 30, 1/4 @ 30 but then you'd have to do the math yourself. This synchronization feature can also be used to do polyrhythms. A three against four polyrhythm can be specified with:

track 0: 3/4 @120, 3/4 @120 track 1: 4/4 , 4/4

Four beats on track 1 are played in the same amount of time that it takes to play three beats on track 0 (1½ seconds).

#n—A # sign followed by a number is a repeat count. For example, you could specify ten measures of 6/8 at metronome 100 with:

track 0: #10 6/8 @100

Note that 10 of the 1,280 measures are being used here. This isn't usually a problem, but if it is, the following command outputs the same number of beats but uses only one measure:

track 0: 60 @100

The earlier, two-measure-long, polyrhythm can be restated as:

track 0: #2 3/4 @120 track 1: #2 4/4

()—A measure surrounded by parentheses is counted silently—no sounds are output during that measure. This command is useful if you need measure-long rests in your music. For example, three measures of 3/4, followed by three measures of a three against four polyrhythm, followed by three more measures of 4/4, can be specified as follows:

track 0: #3 3/4 @100, #3 3/4 @100,

```
Play four measures of 3/4, followed by four measures of a 3/4
 against 4/4 polyrhythm, followed by four measures of 3 against
  4 against 5, followed by four measures of 3 against 4 against 5
 against 7. Sound a warning tone one measure before each change.
track 0: #43/4 a120 w, #43/4 a120 w, #43/4 a120 w, #43/4 a120 w
track 1: (#43/4),
                                      #44/4,
                                                     #44/4
                        #44/4,
track 2: (#43/4),
                       (#45/4),
                                      #45/4.
                                                     #45/4
track 3: (#43/4),
                        (#47/4),
                                      (#47/4).
                                                     #47/4
; Now go into Steve Reich mode. Play 20 measures of 4/4 at
; metronome 150 and at the same time play 20 measure of 4/4
; at metronome 151. The clicks start out on the same
; beat and they very gradually go out of phase and then
; come back into phase again.
track 0: #20 4/4 a 150
track 1: #20 4/4 a 151
```

The C Programmer's Assistant

C TOOLSET

UNIX-like Utilities for Managing C Source Code

Example 1: A click program

No C programmer should be without their assistant—C ToolSet. All of the utility programs are tailored to support the C language, but you can modify them to work with other languages too.

Source code in standard K&R C is included; and you are welcome to use it with any compiler (UNIX compatible) and operating system you choose.

12 Time Savers

DIFF - Compares text files on a line-by-line basis; use CMP for byte-by-byte. Indispensable for showing changes among versions of a program under development.

GREP - Regular expression search. Ideal for finding a procedural call or a variable definition amid a large number of header and source files.

FCHART - Traces the flow of control between the large modules of a program. PP (C Beautifier) - Formats C program files

so they are easier to read.

CUTIL - A general purpose file filter.

Requires MSDOS and 12K RAM

CCREF - Cross references variables used within a program.

CBC (curly brace checker) - checks for pairing of curly braces, parens, quotes, and comments.

Other utilities include DOCMAKE, ASCII, NOCOM, and PRNT.

Source code to every program is included!

Thorough User Support Text and Online

C ToolSet documentation contains descriptions of each program, a listing of program options (if any), and a sample run of the program.

On-line help gives you information on the programs and how to run them. Most of the programs respond to -? on the command line with a list of options.

Call 800-821-2492 to order C ToolSet risk-free for only \$95.

Solution

541 Main Street, Suite 410D So. Weymouth, MA 02190 617-337-6963

CIRCLE 152 ON READER SERVICE CARD

(#3 3/4 @100) track 1: (#3 4/4), #3 4/4 , #3 4/4 You could do the same thing with: track 0: #3 3/4 @100, #3 3/4 @100 track 1: (#3 4/4), #6 4/4

warning—The warning command causes a warning tone (consisting of

two high-pitched beeps) to be played rather than the normal downbeat of a measure.

If the measure is created with a #n command, only the last measure of the series is affected. For example, if you want to do four measures of 3/4, followed by a change to 6/8 but with

a warning just before the change, you'd say:

track 0: #4 3/4 @120 warning,

#10 6/8 @100

The program plays three measures of 3/4 as usual; then it plays a final measure of 3/4, but in this measure the warning tone is used for the

Flotsam and Jetsam

Multiple-Statement Macros

This month's Flotsam and Jetsam discusses how to write macros that have more than one statement in them. I've touched on some of this stuff in previous columns, but it's worthwhile to get everything in one place.

Two-statement macros such as:

```
#define X() stmt1(); stmt2()
```

are not usually a good idea. This macro, when used in:

```
if( condition )
   X( );
else
   something( );

expands as:

if( condition )
   stmt1( );
stmt2( );
else
   something( );
```

Adding curly braces does not solve the problem:

```
#define X() {stmt1(); stmt2();}
```

when used in:

```
if( condition )
   X( );
else
   something( );
expands to:
```

if(condition)
{
 stmt1();
 stmt2();
}
;
else
 something();

The semicolon is a legitimate statement in C, so the *else* tries to bind to the preceding semicolon rather than the *if*.

One workable solution to the problem was used in the program I discussed this month, in which I've used an *if* statement to create a block. The general form is:

```
#define X() \
    if(1) \
    {\
        stmt1(); \
        stmt2(); \
    }else
```

if(condition)

You can even declare variables after the open curly brace if you like (see Listing Two, lines 50–61, page 82, for an example). These variables shadow (take precedence over) any variables that might have the same name in an outer block (they'll be different variables). The trailing, but empty, else clause is required. Without it, the following wouldn't work:

```
X();
else
something();

It would expand as:

if(condition)
if(1)
{
...
```

The extra *else* statement forces the *else something()* to bind properly.

something();

A disadvantage of this method is that an *if* statement is not an expression. Consequently, you can't say $y = X(\cdot)$; because it would evaluate to the

illegal y = if(1)... A way around this problem is to use the comma (or sequence) operator. The comma operator evaluates left to right and evaluates to the rightmost thing in the list. So:

```
#define X() \
    (\
        stmt1(), \
        stmt2() \
)
```

executes stmt1() first, then stmt2(). The entire expression evaluates to the return value of stmt2(). Note that I'm using parentheses, not braces, for grouping.

A disadvantage of the comma operator is that you can't declare variables local to the macro, as you can with braces.

Don't confuse the comma operator with the comma that's used in a subroutine call. They're different things. In order to get a comma operator into a subroutine call, you have to surround the expression with parentheses, as in foo((a,b), c). Here, the left comma is a comma operator, and the one on the right is an argument-list separator. This same caveat applies to the D() macro that I discussed in the November 1986 Flotsam and Jetsam. The preprocessor accepts D(printf("%d",x);) but only because the comma is surrounded by parentheses. The following won't work:

```
D( printf("%d", )
D(x); )
```

because the preprocessor intercepts the trailing comma and looks for a second macro argument. Note that the comma used in a *for* statement, such as for(i=0,j=9;i< j;), is indeed a comma operator.

Think of us as the Book of the Mind Club.

A D V A N C E D

5005

The

Mirrornis

guide for

Assembly

Language

and C

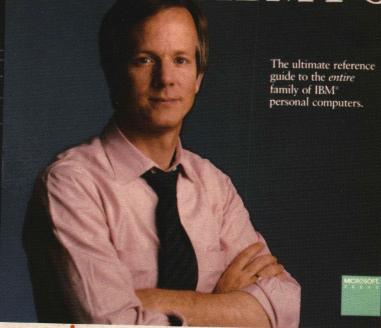
programmers

MICROSOFT QuickBASIC

Developin
Structures
Programs
With
Microsoft's
Advanced
ASIC

Douglas Hergert

MICROSOFT PRESS PROGRAMMER'S GUIDE TO THE IBM PC



PROFICIENT



The Microsoft guide

to advanced C programming.



CIRCLE 125 ON READER SERVICE CARD

Want to turn programming time into prime time? Want to put some topspin on your techniques? Want to develop invaluable new resources?

Time to hit the books. From Microsoft® Press. The best and brightest books in the business.

Our parent company is Microsoft, the folks who taught the PC how to think. Our authors read like a Who's Who of What's What.

Here are four ways to boost your computer's I.Q.:



Advanced MS-DOS* by Ray Duncan. Also known as an information bonanza for assembly language and C programmers. Disk files, records, directories, volume labels, internals, memory management, EXEC functions, installable device drivers. More. Ray Duncan has it down. Now you can, too. \$22.95.

468 pages. Softcover.



The Peter Norton Programmer's Guide to the IBM* PC by Peter Norton. Want to develop intermediate and advanced programs you can port from one branch of the PC tree to another? Want to understand the hardware? Software? The differences between PC, XT, AT and Jr.? Get the latest tech

talk? Relax. The leading authority in the field leads you out of the bog. \$19.95. 448 pages. Softcover.



Microsoft QuickBASIC by Douglas Hergert. Here's the perfect way to get up to speed with QuickBASIC. Plus five, smart, sample programs that'll tweek your QuickBASIC skills: MORTGAGE, for data types; QUICKCHART, for graphics; SURVEY, for data-file techniques; EMPLOYEE, for random-access

files; TWENTY-ONE, for IF...THÉN...ELSE games. \$18.95.384 pages. Softcover.



Proficient C by Augie Hansen.
Cross DOS and C and what do you
get? Powerful programs that run at
warp speed. Use the ANSI.SYS
device driver and the MAKE and
LIB utilities to learn valuable,
reuseable methods of structured
program development. From the
man whose proficiency at Bell
mice and Routheon wes the spring

Labs, General Dynamics and Raytheon was the springboard to this expert guide for intermediates—and experts. \$22.95. 512 pages. Softcover.

Don't fumble for answers. Turn to Microsoft Press. Remember: What you get out of your PC depends on what you read into it.

Available wherever books and software are sold. Credit card orders call 1-800-638-3030. In Maryland call collect, 824-7300.



Microsoft and MS-DOS are registered trademarks of Microsoft Corporation. IBM is a registered trademark of International Business Machines Corporation.

Circle no. 125 on reader service card.

C CHEST (continued from page 113)

downbeat; finally, it plays ten measures of 6/8. That is, you get a warning at the beginning of the measure that immediately precedes the new time signature.

This command can be abbreviated to w or W if you like (actually, any word that starts with a w will do). The frequency of the warning tone is not affected by the -d command-line switch, and it cannot be suppressed with parentheses.

Implementation

Click is implemented in Listings One-Seven, pages 82-96. Listing One, debug.h, is a general-purpose file that contains macros I use regularly. The D() macro was discussed in the November 1986 C Chest (it's also in Bound Volume 11, page 740). Its argument goes away when DEBUG isn't defined. The PUBLIC and PRIVATE macros just declare a subroutine as static or not. They're useful because you can easily change the storage classes of all private subroutines to nonstatic when you're debugging. The UX() and MS() macros work just like the D() macro does. Here the argument to MS() is used only if MS-DOS is defined, and the argument to UX() is used only if MS-DOS isn't defined (UX stands for Unix).

Listing Two presents hardware.h, a catchall file for IBM PC hardware defines. These macros define various numbers used for talking to the counter-timer chip in the PC and for turning the speaker on and off. The process is described in detail in the books listed in the bibliography. Timer channel 0 is used for the system clock interrupt, channel 1 triggers a memory refresh and shouldn't be touched by your program, and channel 3 is used to control the frequency of the PC's speaker.

The SETFRQ macro is interesting because it does more than one thing in a single macro. It's described in depth in this month's Flotsam and Jetsam. You pass it a desired frequency (in Hz), and it causes the bell to ring at that frequency the next time the speaker is enabled (with a SPKR_ON() invocation). The frequencies of the 12 notes in the middle octave of an equal-tempered chro-

matic scale (as on a piano) are defined in notes.h (Listing Three) (C4 is middle C on a piano). Go up an octave by multiplying by 2, down by dividing. Go up a half step (from C to #C, for example) by adding the TWELFTH_ROOT _OF_TWO to a note.

The beep(freq, duration) subroutine (Listing Four) uses all this stuff to ring the bell on the PC at a specified frequency for an indicated number of seconds. Both of these numbers can be fractional. For example:

#define <notes.h> beep(G4, 0.5);

plays the G above middle C for a half second.

The ROM BIOS keeps a running count of clock ticks.

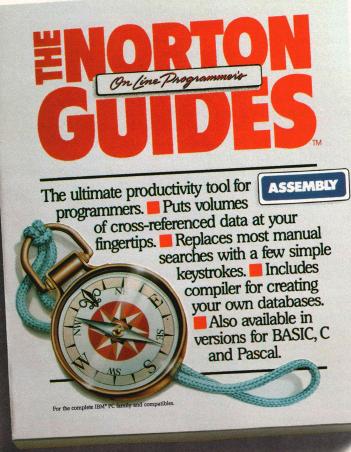
The note duration is determined by the argument sent to the delay() subroutine, called on line 18 and declared in Listing Five. The ROM BIOS keeps a running count of clock ticks, accessible via function 0 of interrupt 0x1a. 0 is midnight on the day that the machine was started, and the interrupt returns with AH set to nonzero when you roll past midnight. Delay() just waits in a tight loop for the correct number of ticks to pass.

Hitherto I've looked at pretty straightforward code, described in most books on DOS programming. With Listing Six, things get more interesting. The IBM PC system clock ticks once every 18.2 seconds (more or less). This resolution isn't adequate for musical applications—for example, click couldn't distinguish between metronome 150 and 152 at the default resolution. Moreover, sitting in a tight loop waiting for the clock to tick isn't a good way to do things—it's too easy to miss a tick.

These problems are solved in two ways. First, you have to speed up the system clock in such a way that you don't mess up the real DOS clock. Second, you have to provide the interrupt service routine to take care of the faster clock tick. If you speed up the clock by a factor of 4, for example, your interrupt service routine is



Peter Norton new programmi who hate



Nobody ever said programming PCs was supposed to be easy.

But does it have to be tedious and time-consuming, too?

Not any more.

Not since the arrival of the remarkable new

program on the left.

Which is designed to save you most of the time you're currently spending searching through the books and manuals on the shelf above.

The Norton On-Line Programmer's Guides™ are a quartet of pop-up reference packages that do the same things in four differ-

the same things in four different languages.

Each package consists of two parts: A memory-resident Instant Access™ program. And a comprehensive, cross-referenced database

crammed with just about everything you need to know to program in your favorite language.

And when we say everything, we mean everything.

Everything from information about language

Designed for the IBM®PC, PC-AT and DOS compatibles. Available at most software



announces a ng tool for people manual labor.

syntax to a variety of tables, including ASCII characters, line drawing characters, error messages, memory usage maps, important data structures and more.

How much more? Well, the databases

for BASIC, C and Pascal give you detailed listings of all built-in and library functions.

While the Assembly database delivers a complete collection of DOS service calls, interrupts and ROM BIOS routines.

You can, of course, find most of this information in the books and manuals on our shelf.

But Peter Norton—who's written a few books himself—figured you'd rather have it on your screen.

In seconds.

In full-screen or moveable half-screen mode. Popping up right next to your work. Right where you need it.



A Guides reference summary screen (shown in blue) pops up on top of the program you're working on (shown in green).

The elect [] No. 803 DO.

| No. 100 [] No. 803 DO.
| No. 100 E. 100 DO.
| No. 100 DO

Summary data expands on command into extensive detail. And you can select from a wide variety of information.

This, you're probably thinking, is precisely the kind of thinking that produced the classic Norton Utilities.™

And you're right.
But even Peter Norton can't think of everything.

Which is why there's a built-in compiler for

creating databases of your own.

And why all Guides databases are compatible with the Instant Access program in your original package.

So you can add more languages without spend-

ing a lot more money.

To get more information, call your dealer. Or call Peter Norton at 1-800-451-0303 Ext. 40.

And ask for some guidance.



dealers, or direct from Peter Norton Computing, Inc., 2210 Wilshire Blvd., #186, Santa Monica, CA 90403. 213-453-2361, Fax 213-453-6398, MCI Mail: PNCI ©1987 Peter Norton Computing

C CHEST

(continued from page 115)

activated on every tick and your routine jumps to the default service routine on every fourth tick. As a final convenience, you want to be able to write your own interrupt service routine in C, without having to go to assembly language.

All of this is done by the routines in Listing Six. The *speedup()* subroutine is called with:

Factor is the speedup factor. Set it to 4 for a fourfold increase in the clock speed. The *funct* argument is a pointer to an interrupt service routine that is called on every clock tick. I'll look at one of these shortly.

Speedup() begins on line 93 of Listing Six. The first thing it does is remember the funct and factor arguments in two local variables: service and tick_reset (declared on lines 41 and 45). It also remembers the value

of the current data segment (on line 103). You need to do this because it's convenient for the C interrupt service routine to access static data and global variables. Because the timer interrupt can come at any time, you have no idea what number is in the DS register when the interrupt is ser-

When the C routine returns, the service routine decides whether or not to call the default timer interrupt.

viced; odds are it is incorrect (it will be the *DS* associated with the interrupted process). By remembering the *DS* register now, you can restore it later. Note that all the variables used by the interrupt service routine are being put into the *_TEXT* segment, which is normally used only for

code. I'm doing this because the contents of the *CS* register must be valid when the service routine is entered (or you couldn't be there). So, you can always get at a variable in the *_TEXT* segment by using a *_TEXT*: or *CS*: segment override. Then you can retrieve the previously stored *DS* contents from a variable in the code segment.

Lines 108 to 124 of Listing Six speed up the clock hardware. Note that you have to test explicitly for a factor of 1 on lines 110 and 111 to avoid an arithmetic overflow exception. In this case, you want to load the counter with the default count of 0, which vields 64K ticks between interrupts (that's the default 18.2 seconds). A speedup(1...) call is useful if you don't want to change the tick rate but do want to install your own service routine. Finally, the routine gets the old timer interrupt (8) vector, using DOS function 0x35 on line 133; saves it in old_seg:old_off; and then installs a new interrupt using DOS function 0x25 (on lines 133-140). You have to push the DS register (on line 136) because function 0x25 is passed the new vector in DS:DX. It's restored on line

You'll note that I didn't actually install the user-supplied service routine in this last step; rather, I installed a pointer to the serv subroutine (lines 156-207). This routine sets up the machine environment so that the C routine can be called safely, and only then does it call the C function. Serv starts out by setting up a new stack (on lines 160-164). A small, 128-byte stack (declared on line 52) is used for this purpose. Then it pushes all the registers onto the new stack (lines 166-173). The old data segment is restored next so that the C function can get at global and local-static variables. Finally, the Croutine is called (on line 179) indirectly through the pointer I set up earlier (on line 102).

When the C routine returns, the service routine decides whether or not to call the default timer interrupt. A running count (numticks) is decremented on each call. When it reaches 0, you reset it to tick_reset (the first argument to speedup()) and then jump to the old service routine (on line 205), again indirectly through a pointer. If numticks hasn't gone to 0, you send a nonspecific EOI (end of in-

IS THERE A VOID IN YOUR LIFE?

Admit it — you've been missing something. What you need is a practical publication that speaks Turbo Pascal, and Turbo Pascal only. *Turbo Tech Report* does just that: this bimonthly newsletter disk publication provides practical programming solutions both on disk and on paper. The approach is hands-on and how-to. Every issue contains:

- Articles written by Turbo Pascal experts.
- Reviews of the latest, hottest Turbo Pascal software products.
- Practical demonstrations of how to solve a particular problem with a Turbo Pascal product.
- A disk filled with code. You'll receive applications developed by authors, plus useful utilities, libraries and routines from Turbo Pascal users worldwide.

Fill the void with a subscription to <u>Turbo Tech Report:</u> 6 issues with 6 disks for \$99. Call (800) 533-4372 or Calif. residents call (800) 356-2002 and start your subscription today!

terrupt) to the hardware (on line 197) and then do an *iret* to terminate the interrupt.

The remainder of Listing Six is straightforward. Slowdown(), defined on lines 213-247, just slows the clock down to its original rate and uninstalls the custom service routine. Slowdown() must be called by your program before it terminates or the machine will go into outer space the next time a timer interrupt occurs. The *cli()* and *sti()* routines on lines 70-78 just disable and enable interrupts from C. They're useful when trying to avoid various communication problems between the interrupt service routine and the rest of the program. You'll see how in a mo-

Now that I've laid the groundwork, I can actually discuss the metronome program. Click.c is presented in Listing Seven. A few of the #defines at the top of the listing are interesting. ROUND(x) takes as input a floatingpoint number (either float or double) and converts it to an int. It rounds to the nearest integer value, however, rather than just truncating-the default behavior of the Microsoft compiler. The various definitions needed to figure the clock rates are on lines 42-46. FACTOR is the speedup() factor. If the clock runs at less than 16 times the default tick rate, the program won't be able to resolve the timing by a single metronome count—it won't be able to differentiate between metronome 150 and 151, for example. DEFAULT_TICK is the default 18.2 times/second used by the system clock. ONE_TICK is the number of times that the clock ticks in a second, given the speedup factor defined earlier. Finally, TICKS(x) converts a metronome x into a number of clock ticks.

The virtual tape is defined with a system of *typedefs* on lines 60–78. Each measure is represented by the *MEASURE* structure defined on lines 63–72. *Num_beats* is the number of beats in the measure, and *ticks_per_beat* is the number of timer ticks between every beat. *Num_beats* is decremented on every beat, and when it reaches 0, the program goes to the next measure. When you're synchronizing with track 0 in a polyrhythmic mode, however, *num_beats* * *ticks_per_beat* is not necessarily the

exact number of clock ticks in the corresponding measure. The remainder field makes up the difference. The extra ticks represented by remainder are spread over the first few beats of the measure to make up any discrepancy. This way the downbeats of synchronized measures will always coincide. Cur_tick is the current clock tick. It is initially set to ticks _per_beat and is decremented on every timer interrupt. When it reaches 0, a tone is output and the variable is reinitialized to ticks_per _beat. Num-_beats is also decremented at this point. The rest of the structure is just housekeeping: silent is set when this measure is silent; warning is set when a warning pulse is to be used on the downbeat of each measure. All these fields are used (and modified) by the interrupt service routine on each clock tick.

A TRACK (on line 75) is an array of MEASURE structures, and the Tape (line 77) is an array of four TRACKS. The Measure array is an array of pointers, one for each track. These pointers point to the current measure on each track. They are incremented by the interrupt service rou-

tine every time it rolls over to a new measure.

The next set of global variables (lines 82-102) are used by the interrupt service routine to pass information to the main body of the program. Ring_bell is set when the bell should be sounded. It is set to one more than the track number or to the special value WARNING, defined on line 56, if the warning tone should be sounded. Collision is set true when there's a collision between two tracks (a beat occurs on both tracks at the same time). Downbeat is incremented on the downbeat of every measure on track 0; it's used to print the measure number to the screen. Done is set to true when all tracks are exhausted, and Numticks is incremented on each timer interrupt (it's useful primarily for debugging).

By far the largest subroutine in the program is *build_tracks()* on lines 219–358, which parses the input file and initializes the *Tape()* array. In spite of its length, it's pretty straightforward and requires little additional comment here.

More important is the interrupt service routine itself, timr_intr() on

Worried about productivity, performance or quality? Find peace of mind...

C Programmer's Toolbox Volumes I & II

23 powerful tools for the IBM PC, AT and compatibles, developed by professionals with many years of experience in system and application software development. For both beginning and experienced programmers.

- Monitor program/system execution
- Beautify program listings
- Determine program flow/critical paths
- Trace and verify variable usage
- Filter input/output streams
- Create/modify/verify file contents
- o Powerful, common user interface
- o Interactive and batch execution
- o Online documentation
- o Self-explanatory error messages

At \$79.95 per volume or \$130 for both with a 30 day trial, the C Programmer's Toolbox is simply the best value today. In addition there are more than 200 pages of documentation, packed with examples and tips on creating better programs.

Why waste time, call or write us today.





MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781

PRESENTING THE DIFFERENCE BETWEEN FAST COMPILING AND FAST PROGRAMMING.

For compiling speed, you can't do better than Let's C. But to really speed up programming you can't do without the powerful source level debugger, csd.

If you want the power, portability and flexibility of C, start with the complete compiler, Let's C. For utilities, editor, compiling speed and fast, dense code, Let's Chas it all.

But to get your programs up and running you

need more. Because even the fastest compiler can't outrun bugs. You need the revolutionary C Source Debugger, csd.

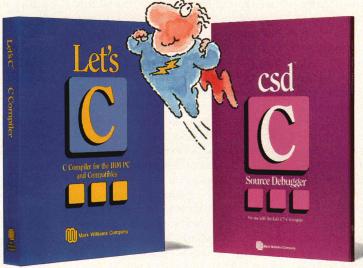
CUT DEVELOPMENT TIME IN HALF WITH csd

csd lets you bypass the time consuming frustrations of debugging-like long dumps and clunky assembler. With csd, you actually debug in C. You learn faster because you watch your program run in C. You finish faster because csd combines the speed of a compiler with the interactive advantages of an interpreter. The end result? Development time is sliced in half.

LET'S C AND csd FEATURES

Let's C:

- Integrated edit-compile cycle: edi-
- · Includes both small and large
- Integrated environment or
- lexicon format
- New make utility
- extensions
- make, assembler, archiver, cc onestep compiling, egrep, pr, tail, wc
- MicroEMACS full screen editor with source included
- party libraries



LIMITED TIME

OFFER

FREE csd

WITH LET'S C!

REVIEWERS ARE RAVING ABOUT LET'S C AND csd.

"Let's C is an inexpensive, high-quality programming package... with all the tools you will need to create applications." -William G. Wong, BYTE. August 1986.

"The performance and documentation of the \$75 Let's C compiler rival those of C compilers for the PC currently being sold for

\$500...highly recommended..."

-Marty Franz, PC TECH JOURNAL, August 1986.

"csd is close to the ideal debugging environment...a definite aid to learning C and an indispensable tool for program development." -William G. Wong, BYTE, August 1986.

"This is a powerful and sophisticated debugger built on a well-designed, 'serious' combiler."

-Jonathon Sachs, Micro/Systems Journal, April, 1986

START TO FINISH, THERE'S NO BETTER ENVIRONMENT.

Get started with the right C compiler and you'll have everything you need for development-including source level debugging. On top of it all, Let's C and csd are today's best values in professional C programming tools. And most reliable: Mark Williams C compilers have been sold with DEC, Intel and Wang computers since 1981.

60 DAY MONEY BACK GUARANTEE

Mark Williams gives you a full 60 days to find out just how good Let's C and csd really are—or your money back.

So if you want more than a fast compiler—if you want your programs up and running fast, ask for Let's C and csd. You'll find them at your software dealer's, in the software department of your favorite bookstore, through the Express Program at over 5500 Radio Shacks or you can order now by calling 1-800-MWC-1700.* *In Illinois call, 1-312-472-6659.



1430 West Wrightwood, Chicago, Illinois 60614

© 1987 Mark Williams Company Let's C is a registered trademark of the Mark Williams Company. UNIX is a trademark of Bell Labs.

- Now compiles twice as fast
- tor automatically points
- memory model
- command line interface
- 8087 sensing and support Documentation features new
- MS-DOS object compatible
- Fast compact code plus register
- Full Kernighan & Ritchie C and
- Full UNIX compatibility and complete libraries
- Many powerful utilities including
- Supported by dozens of third

- · For the IBM-PC and Compatibles
- Not copy protected

Sieve Benchmark

(Compile time in seconds) Let's C: 2.8 (On 512K 6Mhz IBM-AT) Turbo C: 3.89 (As advertised)

csd:

- Large and small memory model
- Debug in C source code, not assembler
- Monitor variables while tracing program
- · Does not change program speed
- · Provides separate source, evaluation, program and history windows
- On-line help screens
- · Can interactively evaluate any C expression
- · Can execute any C function in your program
- Trace back function
- · Ability to set trace points Not copy protected
- MARK WILLIAMS LET'S C AND csd. ONLY \$75 EACH.

MICRO/ SYSTEMS JOURNAL

FOR THE ADVANCED COMPUTER USER

SUBSCRIBE NOW AND

SAVE OVER 15% OFF THE

OFF THE NEWSSTAND PRICE!

	Yes! I want to subscribe to Micro/Systems Journal.
ind sav	ve over 15 % off the cover price.
1 Y	ear (6 issues) \$20 🔲 2 Years (12 issues) \$3
Please ch	arge my: Uisa MasterCard American Express
] Payment	enclosed
ard #	Exp. date
ignature	
ame	
ity	StateZip
ear. All count	on a full one-year cover price of \$23.70. Canada and Mexico add \$3 for surface mail, \$7 for airmail per ries add \$12 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. llow 6-8 weeks for delivery of first issue.
	A Publication of M & T Publishing, Inc.
	0.
	Yes! I want to subscribe to Micro/Systems Journal
	Micro/Systems Journal
and sa	ve over 15% off the cover price.
7 1 Y	ear (6 issues) \$20 🔲 2 Years (12 issues) \$3
Please cl	harge my: Usa MasterCard American Express
Paymen	t enclosed Bill me later
Card #	Exp. date
Signature _	
Name	
Address _	
	StateZip
year. All coun	on a full one-year cover price of \$23.70. Canada and Mexico add \$3 for surface mail, \$7 for airmail per tries add \$12 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S.
oank. Please	allow 6-8 weeks for delivery of first issue. A Publication of M & T Publishing, Inc.
	Yes! I want to subscribe to Micro/Systems Journal
1	Yes! I want to subscribe to
	165: I Wallt to Substitute to
	Micro/Systems Journal
and ca	ve over 15 % off the cover price.
	Year (6 issues) \$20 🔲 2 Years (12 issues) \$3
☐ Please	charge my: Uisa MasterCard Marcican Expres
☐ Payme	nt enclosed Bill me later
	Exp. date
Cianata	
Name	
Name	State Zip



FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

For the Advanced Computer User

Micro/Systems Journal.

Box 3713 Escondido, CA 92025-9843

No Postage Necessary If Mailed In The United States





BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

For the Advanced Computer User

Micro/Systems Journal

Box 3713 Escondidio, CA 92025-9843

Halandalllaandalalalalalalalalalalalalal

No Postage Necessary If Mailed In The United States



BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

For the Advanced Computer User

Micro/Systems Journal

Box 3713 Escondidio, CA 92025-9843

Ունուսի Միուսի հեն հեն հունոնուն հեն հեն հ

No Postage Necessary If Mailed In The United States



(continued from page 119)

lines 362-427. Though the routine is just a C subroutine, there are several issues that must be kept in mind while writing it. First, MS-DOS is not reentrant. In practice, this failing means that you can't call most DOS functions from an interrupt service routine (because the program might be in DOS when it was interrupted). The second issue is return values. Because the service routine isn't called in the normal way, it can't be passed parameters and it can't return a value in the normal way. Global variables must be used for this purpose. The remaining problems are stackrelated. The interrupt service routine uses its own stack, so you have to disable the default stack-overflow checking that's inserted by the compiler (which will almost always fail because the stack isn't where its supposed to be). This disabling is often done with a compiler command-line switch, but the Microsoft compiler lets you do it with the #pragma check _stack directives (on lines 362 and 427). The #pragma lets you disable the checking for one subroutine only, instead of affecting the entire module. A trailing minus sign disables checking and a plus sign enables it. The final stack issue is its size. The service routine uses a 128-byte stack, of which 18 bytes are used to save registers and do the subroutine call. You have to be careful not only about the amount of stack used by your own local variables but also the amount of stack used by subroutines that the service routine might call. Be careful. If you need more stack, change the declaration for stack on line 52 of Listing Six.

The service routine itself modifies the global variables described earlier. The *for* loop is executed four times—one iteration for each track. The test on line 377 is for the end of track. If the beat count is 0, there are no more measures on this track. The next test (on line 379) checks to see if the bell should sound (this is the case if the current count is the maximum). The current tick is then decremented on line 402, and if it goes to 0, the number of beats is also decremented (on line 404). The routine advances to the next measure, if necessary, on line

409. The else clauses on lines 411–421 takes care of the Remainder—the extra clock ticks that have to be inserted to keep synchronized with track 0. I add 1 to the current tick count on line 418 to do this. That is, I stretch the current beat out by one clock tick. This way the extra ticks are spread over the first few beats in the measure instead of being piled in one place.

The main() subroutine has to do several things to get the ball rolling. First, it must call signal() to guarantee that slowdown() is called if you abort the program with Ctrl-Break. Signal() is called on line 495, and it installs the on_break() subroutine (lines 431-442) to call slowdown(). Speedup() is called in the initialization part of the for loop on line 497. The loop terminates when the interrupt service routine says that it's finished by setting the Done flag. The code in the body of the loop just tests to see if it should ring the bell and rings it if necessary. I'm doing this here rather than in the service routine because it's easier. The problem is the delay between turning the bell on and then off again. You can't wait in the service routine itself because the timing would be thrown off. By waiting here, the wait time is essentially independent from the start-ofbeat timing. Of course, you could turn the bell on in the service routine and then set a flag, decrementing the flag on each timer tick and turning the bell off when the flag gets to 0. This method seems a lot of bother, however, so I took the easy way out. The final point worth mentioning is that interrupts have to be disabled while you do the tests because there are two variables involved and you don't want one of these to magically change its value halfway through the test. Disabling interrupts prevents this change.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 336-3600, ext. 216. Please specify issue number and format (MS-DOS, Macintosh, Kaypro).

Because the code this month is pretty compiler dependent, I'm distributing executable versions (along with the full source code) through Software Engineering Consultants, P.O. Box 5679, Berkeley, CA 94705. This version has been enhanced to allow several different kinds of warning tones and to allow duration to be specified in minutes and seconds as well as beats. The cost is \$20.

Bibliography

Brickner, Ralph G. "An Execution Profiler for the PC." *PC Tech Journal* 4:11 (November 1986): 120–142. This article describes another program that steals the system timer interrupt; the code of interest is in Listing 2, pages 140–142.

IBM Technical Reference. The BIOS listing contains the default timer interrupt service routine. It's on page 5-162 of the AT technical reference and on page A-79 of the XT reference. The routine is called TIMER_INT in both listings.

Norton, Peter. The Peter Norton Programmer's Guide to the IBM PC. Bellevue, Wash.: Microsoft Press, 1985. This book contains information on how the bell on the IBM PC works as well as information on the other DOS interrupts used by click. The information is duplicated in innumerable other books about programming the IBM PC.

DDJ

(Listings begin on page 82.)

Vote for your favorite feature/article. Circle Reader Service **No.** 5.

V.I.P., Clustered Binary Trees, and Clustered List Data Structures

This month I discuss the Visual Interactive Programming (V.I.P.) language, a new icon-based interpreter for the Apple Macintosh computer. My second topic conforms with this issue's theme—algorithms—by presenting modified structures for binary trees and linked lists.

The Macintosh has been blessed with numerous language interpreters and compilers: BASIC, FORTRAN, Pascal, Modula-2, C, LISP, PROLOG, Forth, and so on. Now, Mainstay of Agoura Hills, California, has developed a new language that truly takes advantage of icons and the Macintosh user interface. V.I.P. is an interpreter that breathes life into a flow-chart—instead of typing text for the source code, you can assemble a program using special flowchartlike symbols.

V.I.P. also incorporates a programming environment. Its appearance resembles MacPaint: a menu bar across the top; a flowchart viewing port; and to the left of this port, three groups of icons—object (data-type) icons, icons for loops and decisionmaking constructs, and icons for several classes of predefined routines. To build a program you point to an icon with the mouse and click. The environment's response takes one of two forms: a window or a flowchart icon. The window form is fairly typical of the Macintosh and involves a more sophisticated level of interaction with the user. The icon form lets you fully define the flowchart sym-

by Namir Clement Shammas

bol and represents a simpler level of interaction. Each flowchart symbol has a comment line at the bottom.

The product uses the same geometric shape—namely, a horizontal rectangular strip—for all the icons. At



the two edges of the rectangular icon are two squares: one to open the icon, the other to close it. Figure 1, page 123, shows a sketch of an opened FOR-NEXT icon. The rectangle containing the for keyword is the permanently visible part of the icon. The two columns below it are for input or inspection. The left column of rectangles labels the information required and encloses the object type in parentheses. V.I.P. uses lowercase and uppercase letters for the objecttype codes to indicate input and output, respectively. The last row is reserved for comments.

V.I.P. supports a fixed set of object (data and constant) types. They are:

- BYTE, which uses 1 byte of storage. Short integers (-128 to + 127) or single characters can use this type.
- *INTEGER*, which uses 4 bytes to accommodate long integers (between minus and plus 2 billion).
- *REAL*, which occupies 10 bytes with a 64-bit mantissa and 15-bit exponent. Thus, floating points with 19 significant figures are supported with an exponent varying from -4,932 to +4,932.
- *POINT*, which requires 4 bytes of storage to represent a vertical and horizontal set of coordinates. (The range of values for each axis is –32,678 to +32,767.) This is equivalent to a predefined record structure in a structured language such as Pascal or C.
- RECTANGLE, which uses 8 bytes to represent four integers that define the upper-left and lower-right cor-

ners of a rectangle.

• CONSTANT, which is used to implement symbolic constants. V.I.P. supports four types of constants: character, integer, real, and string. The true, false, e, and pi constants are predefined.

Arrays of up to three dimensions are supported, with the lower array bound fixed as 1. When you select a data-type icon, a window directory opens to display a list of all the variables of the selected type. The window lets you choose between global and local variables. You can also insert new variables, delete or change existing definitions, rename variables (V.I.P. is case sensitive), convert scalar ones into arrays and vice versa, and alter array sizes and the number of dimensions.

V.I.P. offers two decision-making constructs: IF and CASE statements. The *IF* statement comes in the form of the IF-THEN-ELSE icon, with two flowchart branches. You can simulate an ELSEIF by using nested IF statements. The IF icon is defined by entering a logical expression. The CASE statement supports up to 30 cases. When you choose the CASE icon, you are prompted for the number of cases to create an icon with the correct number of alternatives. You enter the selector (switch) variable and the values associated with each CASE clause, V.I.P. has no CASE ELSE clause.

The language supports two loop constructs: FOR...NEXT and WHILE...DO. When you use a FOR loop, you must specify the control variable as well as its initial, final, and step values. To insert a WHILE loop icon, you need to specify the logical expression used to iterate the loop.

V.I.P. lets you divide your task into smaller routines—a feature useful for maintaining a clear set of flowcharts instead of using one big flowchart. You can choose to invoke a routine's option from the top menu bar. This opens a new window in which you declare all your routines. You can easily switch between the main routine and any other routine to edit or inspect any program components. Each routine has its own workspace. When you select a routine for the first time, you start with a clear flowchart.

User-defined routines in V.I.P. can take parameters. A special window for parameters opens when you select the "Set argument ..." option. The parameter window resembles the one for declaring variables, with a few differences. First, you must indicate the type and the input/output status. V.I.P. does not allow the same parameter to pass data back and forth between the routine and its caller. You can define the parameters as scalar or arrays (with up to three dimensions). Parameters can be inserted, deleted, or altered. V.I.P. maintains control over parameter passing and verifies that the arguments in the calls correspond to the parameters' type, number, and sequence.

V.I.P. has a versatile set of intrinsic functions. Mathematical functions include square root, logarithmic, trigonometric (and their inverses), and hyperbolic (and their inverses). Other mathematical functions include the absolute value, sign, modulus, minimum, maximum, and random-number generator. Another collection of functions returns the vertical and horizontal coordinates of a point and the four coordinates used to define a rectangle.

The software comes with powerful group of predefined routines. Each class of routines is represented by an icon located on the left-hand side of the flowchart viewing port. There are 15 classes of routines, listed along with their functions in Table 1, right.

The V.I.P. editor enables to you perform search/replace operations on the contents of the flowchart icons and also lets you cut, copy, paste, and delete icons. You can view the flowchart using a smaller scale that can be magnified by pressing the shift key and the mouse button simultaneously. In this mode, however, the screen displays uncommented icons,

which I find annoying.

Finally, V.I.P. includes a versatile debugger that lets you single-step through your flowchart icons, set and remove breakpoints, and set/examine objects.

I ran some of the V.I.P. demonstration programs, which illustrate how easy it is to write programs that use the Macintosh interface. I also wrote two versions of the Sieve benchmark program. The first (see the text version in Example 1, page 124) is written in a typical form—that is, no local routines are used. One iteration of this program took 3 minutes and 35 seconds. Example 2, page 125, shows the text of the second version, which uses two subroutines—*init* and *body*. To the best of my knowledge, V.I.P.

- 1. Assignment: includes simple assignment and filling/copying bytes.
- Mathematics: includes integer, fraction, power, simple financial calculations, and sorting numbers.
- String manipulation: to carry out typical related operations (string concatenation, comparison, append, copying, and conversions with numbers). Sorting an array of strings is also included.
- Graphics: a large set of routines that lets you obtain the best of the Macintosh graphics.
- 5. Event trapping: to inspect the status of the mouse.
- Menu management: to create, enable, disable, remove, and load menus, to name a few.
- 7. Window management: to create, set up, load, remove, and activate windows.
- 8. Text editing: to perform text editing functions, such as copying, pasting, cutting, clearing, and inserting text. Text file I/O operations are also included.
- 9. Dialog management: to permit your programs to display Macintosh dialog windows.
- 10. Sound effects: to play tune, set voice, set notes, and turn sound on/off.
- 11. Record management: to support dynamic record allocation, copying of records, and field I/O.
- 12. I/O operations: a set of I/O routines for file manipulation (open, close, get file position, and so on) and I/O of objects, records, text, and pictures.
- 13. Printing: to set up the printed page and print text and pictures.
- 14. Branching: to exit from loops and execute another program.
- 15. Date and time management: to wait, read the clock, and obtain the time and the date.

Table 1: V.I.P.'s classes of routines and their functions

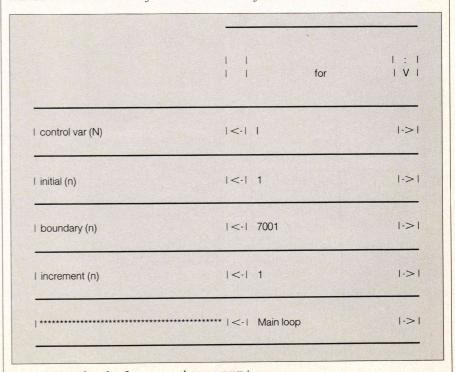


Figure 1: Sketch of an opened FOR-NEXT icon

STRUCTURED PROGRAMMING (continued from page 123)

is the first commercially distributed program of its kind for microcomputers. It offers several interesting programming features-in particular, visual programming, which is excellent for teaching because students don't look at dummy flowcharts but at active ones. Visual programming may also prove valuable for algorithm design. In addition, V.I.P.'s library of routines make it a very capable language. I applaud Mainstay for its efforts and creativity in developing V.I.P. and look forward to more powerful versions. Mainstay has also announced that it will be realeasing V.I.P. translators for Pascal and C.

I have praised both V.I.P. and the concept of visual programming; now let me mention a few shortcomings. First, programmers who type quickly and use keyboard macros may find software development in visual programming languages slow. Second, V.I.P. does not provide an escape mechanism once a flowchart icon is opened. Third, viewing large flowcharts is cumbersome. Fourth, programmers have to inspect V.I.P.'s flowchart icons in order to view their contents. And finally, V.I.P. makes no provision for user-defined objects.

Clustered Binary Trees

Binary trees are useful data structures for internal sorting and searching. Yet, despite all the praise binary trees receive, they are vulnerable to becoming unbalanced. The ultimate nightmare is to input a perfectly sorted array into a binary tree and end up with one long linked list! Two decades ago, two Russian mathematicians devised a few algorithms to maintain a binary tree in a near-perfect balanced state. This type of binary tree is known as an AVL tree. Yet both the basic binary tree and the AVL tree typically use less-than comparisons to insert a new node or to search for one-they take no advantage of the difference between the values of a resident node and an incoming data item. With binary and AVL trees, you compare the value of a data item with that of a node. If it is less or equal, you use the left node pointer; otherwise, you use the right node pointer.

I suggest the following modification to the binary tree. In the new tree, each node has two left pointers and two right pointers, and so I call it the clustered binary tree (CB tree for short) and name the pointers low left, high left, low right, and high right. Why duplicate the pointers for each side? The answer lies in being able to select the proper pointer to follow, based on the difference between the values of the node key and the incoming data item. A critical difference value (CDV) is preassigned based on the nature of the data. When handling numeric keys, the values of the CDV and the whole algorithm are easy to implement. If the calculated difference is greater than the value of the CDV, then the highleft or high-right pointers are used, depending on the sign of the difference; otherwise, the low pointers are

The use of these four pointers in the CB tree helps to separate nodes with keys that vary widely and so speeds up tree insertion and searching. I used the Turbo Pascal program shown in Listing One, page 98, to compare the speed of insertion and searching in binary and CB trees. The program creates an array of numbers by using one of three methods: random-number generation, the sine function, or the cosine function. Although the speed of searching was about the same, the CB tree was four to five times faster in inserting new data. I used the sine and cosine func-

```
byte
Ch
integer
Count
Diff
Flag[7001]
Iter
Prime
T1
T2
constants
SIZE = 7001
main
read clock (T1)
for (Iter, 1, 1, 1) ** Main loop **
        assign (0, Count)
         for (I,1,SIZE,1) ** Init. flags **
                 assign (1,Flag[I])
         for (I, 1, SIZE, 1)
                 if (Flag[I] = 1)
                          assign (I+I+3, Prime)
                          assign (I + Prime, K)
                          while (K SIZE)
                                   assign (0,Flag[K])
                                   assign (K+Prime, K)
                          assign (Count+1, Count)
                 else
read clock (T2)
wait (5000)
assign ((T2-T1)/60, Diff)
write output (1, "@i", Diff)
get key (100, Ch)
```

Example 1: V.I.P. text output for the first version of the Sieve program

tions to generate arrays with a certain degree of order, which is the weak point of the binary tree.

If string-type keys are used with CB trees, then the ASCII code numbers of the first few leading characters are utilized. You use the Pascal ORD() function to do this. The numeric values used for critical difference calculations are ORD(Key[1]) for using the first character in the key, and:

ORD(Key[1]) * 100 + ORD(Key(2))

for using the first two characters.

Clustered List Structures

You can also use the concept of the clustered binary tree structure with lists. Basically, a clustered single-link list (C list for short) is a list with two sets of pointers: one set forms the "high-track," or fast search lane; the other forms the "low track," or a clustered sublist. Thus, a C list structure has one high-track link and many clustered sublists, each linked to a high-track node, and so a hightrack node becomes an index that indicates the range of data stored in its linked sublist. The effect of the two sets of pointers is best visualized by the two lanes of a highway, in which one is for faster traffic. You normally take the fast lane, as long as you are relatively far from the exit you seek. As you get closer to your exit, you switch to the slower lane. The same idea applies to C lists. By using numerical differences in comparing a node with an incoming datum, you can decide whether or not to use the high-track pointers and so bypass many unnecessary comparisons. This approach makes C lists more efficient in searches than are normal lists. The price you pay is the extra set of pointers.

Listing Two, page 102, shows a Turbo Pascal program that demonstrates C list insertions and displays. The program contains procedures for searching, inserting, and visiting Clists. Notice the following aspects of the program:

1. A string-type key is used. The entire key of any node and incoming data is used in a logical comparison. The numeric difference is calculated for the first character only, however.

2. A CDV of 0 is used, which causes the high-track node to index on a range of one character. The range of char-

number of one key character is used. 3. The Search_Node procedure uses a Boolean flag, HiTrack, to switch acters is (CDV+1) when the ASCII | from using high-track pointers to

```
byte
Ch
integer
Count
Diff
Flag[7001]
Iter
T1
T2
main
read clock (T1)
for (Iter, 1, 1, 1)
        init ** Initialize flags **
        body (Count) ** Body of sieve **
read clock (T2)
assign (T2 - T1, Diff)
write output (1,"@i",Diff)
get key (5000,Ch)
body (Count)
<-- integer Count
integer
Flags[7001]
K
Prime
assign (0,Count)
for (I, 1, 7001, 1)
         if (Flag[I] = 1)
                  assign (I+I+3, Prime)
                  assign (I+Prime, K)
                  while (K 7001)
                           assign (0,Flag[K])
                           assign (K+Prime, K)
                  assign (Count+1, Count)
         else
init
integer
for (I, 1, 7001, 1)
         assign (1,Flag[I])
```

Example 2: V.I.P. text output for the modified Sieve program



NUMERICAL SOFTWARE TOOLS IN C Jim Kempf, Hewlett-Packard Laboratories Paper (013–627274–6) \$26.67

This innovative volume focuses on the use of software tools in designing reusable numerical software using the powerful C programming language. It packages numerical functions as discrete programs or program modules designed to work together. The user interface is kept simple and an emphasis is placed on the techniques of good software engineering.

SYSTEMS SOFTWARE TOOLS Ted J. Biggerstaff, Microelectronics Computer Corporation

Paper (013–881764–2) \$19.95, Cloth (013–881772–3) \$31.00

Build powerful systems software with this handy guidebook. Ted Biggerstaff merges the power of C with the accessibility of the IBM® PC and gives you practical, complete tools to create windowing, multitasking, interrupts and communications capabilities.

A SOFTWARE TOOLS SAMPLER, 1987 Webb Miller, The Pennsylvania State University, Paper (013–822305–x) \$26.67

Build more interesting and sophisticated programs with this handy collection of software tools in C. This book offers tools for file updating and comparison, pattern matching, and a screen editor to help you create a more pleasant and productive programming environment. By the author of Engineering of Numerical Software (Prentice Hall, 1985).

All trademarks and registered trademarks are copyrighted and protected by their respective manufacturers.



CRAFTING C TOOLS FOR THE IBM PCs, 1986 Joe Campbell, Paper (013-188418-2) \$25.95

Eliminate hundreds of hours of research and development in writing applications software for the IBM® PC in C and learn something new, useful, and interesting. This much-needed book presents information on the interfacing of a high-level language to an assemble, an area of programming most programmers need from time to time, but, one on which they have little information. Learn how to implement and manage interrupts within a C program, direct video access, serial communications and directory manipulation. When it comes to "C", this book has it all!

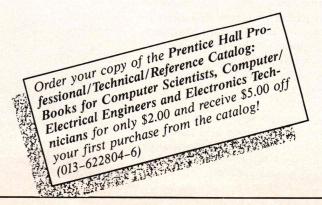
SYSTEMS PROGRAMMING FOR SMALL COM-PUTERS

David H. Marcellus, State University of New York at Binghamton, Paper (013–881656–5)
\$21.95, Cloth (013–881664–6) \$36.00

A practical analysis of traditional systems programs, this book shows readers how to construct editors, interpreters, compilers, operating systems and other systems software for microcomputers. Using a compiler and an interpreter designed for the language TINY BASIC and the CP/M and UNIX® operating systems as examples, readers are introduced to systems programming on microcomputer systems ranging in complexity from a single board micro to multi-user systems.

Available at better bookstores or direct from Prentice Hall at (201) 767–5937. For quantity orders call (201) 592–2498.

Prices subject to change without notice.



STRUCTURED PROGRAMMING (continued from page 125)

low-track pointers during a search.
4. A new datum can be inserted in one of the following locations:

- · as the new head of the entire list
- · in a clustered sublist
- · as a new high-track node
- as the new member of a high-track node, pushing the previous one inside the clustered sublist

The unused high pointers in a sublist node can be used to form a doubly linked sublist. I feel the impact of C lists on list structures is greater than that of CB trees on binary trees. The improved performance brought by C lists is less affected by the type and variation of data than is the case with CB trees.

Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listings begin on page 98.)

Vote for your favorite feature/article. Circle Reader Service **No. 6**.

Easy to C

is a great programming language. Now the C WORKSHOP makes it easy.

Interactive software teaches you C. When you complete and run a program exercise, the amazing Soft Tutor™ gives you immediate feedback, pinpointing any incorrect results.

If you've never programmed before, start with the basic videas of structured programming. Study what you want when you want, including advanced pointer techniques and linked lists.

The C WORKSHOP has everything you need to learn and use C. You can write your own programs, too. The integrated editor and 5500 line/minute compiler are complete with popup menus, customizable keys, online help and C reference lookup.

Let the other guy struggle with confusing books and compilers. Join AT&T and other major



companies now using C WORKSHOP. Columnist Adam Green calls it "the most intriguing new type of training system I've ever seen." (InfoWorld, 1/27/86)

Order your *C WORKSHOP* today. And C how easy it is.

Quality software since 1981

Specifications

Package includes tutorial, Soft Tutor, editor, and C compiler. You get unprotected diskettes and coordinated 384-page book.

Tutorial: Quizzes, exercises, electronic index and bookmarks.

Editor: Search and replace, block commands, split screen, context sensitive C help. Creates ASCII files.

Compiler: Full "K&R" C Addre plus signed char, unsigned long, and re-usable member names. Produces 8086 code. Compiles over 5,500 lines per minute (8 MHz AT). Creates .COM programs. Cursor placed at first error message. Library includes disk I/O, cursor control, printf, scanf, longimp.

Soft Tutor: Detects incorrect output from program exercises. Shows example of the problem. Operates after compiler checks syntax.

Memory usage: Uses 220K. Uses additional RAM as available. If not satisfied, tell us why and return in 30 days for your money back.

Name
Address
City
State
CWorkshop software and book
Ship (we use Priority Mail)
Sales tax in CA (4.90)
Sales tax i

Smalltalk/V

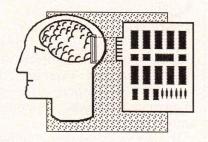
igitalk's Smalltalk/V programming tool is a bit-mapped implementation of a substantial subset of Smalltalk. It is aimed primarily at the AI development market and is code compatible with the earlier Methods product, also by Digitalk. The feature that makes it suitable for AI, aside from the object orientation, is primarily the inclusion of a surprisingly complete and robust PROLOG compiler. This PROLOG includes many predicates that are lacking in Turbo Prolog, for example, such as functor and univ.

Smalltalk/V also has excellent graphics capabilities, such as turtle graphics, and offers good performance in graphics animation. Also included with the product is a large ondisk tutorial that provides some substantial program examples. Digitalk's Methods was the first Smalltalk implementation for PCs, and it was an important landmark as far as many programmers were concerned. But here we have a major subset of it running in about half a megabyte.

The first thing to realize about Smalltalk/V is that it is not just a programming language, but as any Smalltalk should be, it is a full, multiwindow environment with drop-down menus, mouse support, and more conveniences than you have probably ever seen in a programming environment on a PC. A

by Ernest R. Tello

mouse is not necessarily required because Digitalk has implemented an ingenious use of the keyboard using two main clusters of keys on the far left and far right, which it calls the left-hand mouse and right-hand mouse, respectively. If you choose not to use a mouse, you will find that after a while this works quite well.



You can keep both hands in their place and reach for one of the keys in either cluster just as if you were pressing the buttons on two mice. The product was intended to be used with a mouse, however, and works much better with one, if for no other reason than that it makes the cursor fly across the screen instead of crawl. At this time, Smalltalk/V supports the Microsoft mouse and the Mouse Systems mouse. I have also had no trouble using it with the Logitech mouse.

The current version of Smalltalk/V comes on three disks—the Image, Source, and Tutorial disks—and requires 512K. It is a considerable accomplishment to have implemented so much of Smalltalk-80 in half a megabyte.

The dialect of Smalltalk/V is so close to Smalltalk-80 that most of the classes and examples in the Smalltalk-80 book series can be entered "as is." The main exceptions are those that make use of multitasking, such as the simulation examples—the system accepts even these, although they won't work as written. This is important for programmers new to Smalltalk because there is really very little published material available other than what is available for Smalltalk-80 to give them a full overview of the Smalltalk system and help them get going with it.

In the Class Hierarchy Browser, the lower classes in the hierarchy beneath those that are the immediate subclasses of *Object*, the root class, can be either hidden or visible as you choose. Once you have chosen a particular class with subclasses, you can choose to show or conceal just the subclasses under it. Smalltalk/V also has some additional commands on the desktop—for example, now you can cycle around to other windows from a command on one of the dropdown menus. This was needed because, when a window is completely covered by another window, you cannot select it with the mouse.

The basic types of facilities you use with Smalltalk/V are things such as workspace windows, browsers, menus, and occasionally what are known as inspectors. The main types of browsers are Class Browsers, Class Hierarchy Browsers, and the Disk Browser. These are specialized window facilities that give you a viewpoint onto a particular aspect of the system. And as I mentioned earlier, you can create as many instances of these views as you may need.

Class Consciousness

A Class Hierarchy Browser is the Digitalk version of the Smalltalk System Browser. As implemented in Smalltalk/V, this type of browser has four separate panels. The first is a scrolling window that lists the main classes and subclasses in the system. To the right of it is the methods pane, which displays the list of applicable methods. Beneath it are two small selector panes containing the words class and instance, respectively. Finally, on the bottom is a large pane that displays the actual Smalltalk source code for selected items. Depending upon whether you select on the instance or class pane, either the calling names of instance or class methods are displayed in the methods pane. When you select one of these method names, its source code is displayed in the lower pane. When the source pane is current, it acts as a

What experts are saying about PC Scheme from Texas Instruments:

"Texas Instruments has produced an outstanding implementation of a language that could well become as indispensable as C is today."

AI Expert, February 1987

Discover how powerful—and inexpensive—PC symbolic programming can be with PC Scheme from Texas Instruments. Whether you're an experienced Lisp programmer or just beginning, PC Scheme is the complete, \$95* solution to your software development needs.

PC Scheme combines elegant simplicity with remarkable speed in a full Lisp development system. Named PC Tech Journal's Product of the Month (August 1986), PC Scheme brings professional Lisp programming features to personal computers.

PC Scheme 3.0

- Optimizing incremental byte-code compiler for ease of programming and operation
- —EMACS-like editor
- —Lexical scoping of variables
- Ability to suspend PC Scheme, execute DOS-based programs, then return to PC Scheme
- -Random-file access and binary-file support
- Extensions for debugging, graphics and windowing
- External language interface to C,
 Turbo Pascal[®] and other languages
- —SCOOPS (Scheme Object-Oriented Programming System)
- Two-megabyte extended/expanded memory support
- New manuals with tutorials and examples

Find out for yourself why experts are praising PC Scheme. For the dealer nearest you, or to order by phone, call toll-free:

1-800-527-3500

* TI Suggested list price

PC Scheme runs on IBM® Personal Computers and compatibles (including the Texas Instruments Business-ProTM computer). Minimum configuration: 512K RAM, dual floppy system.

Tirbo Pascal is a registered trademark of Borland International. IBM is a registered trademark of International Business Machines Corporation. Business-Pro is a trademark of Texas Instruments Incorporated.



ARTIFICIAL INTELLIGENCE (continued from page 128)

text editing pane in which you can create and modify source code.

What this type of browser means to a software developer is that you have a built-in overview of the system that can give you ready access to anything in the system at all times. The way you generally call upon things that have already been entered into the system is by creating instances of a class and initializing variables in a workspace window and then sending messages to it. Any involved interaction can itself be

made into a program by adding it as a new subclass with its own variables and methods that can be instantiated and evaluated more easily.

As mentioned earlier, Smalltalk/V also provides inspectors. These are special-purpose windows you can use as low-level debugging tools that allow you to examine and even change objects in the system. You don't open an Inspector window by accessing a menu; instead, you use a workspace or system transcript window to send the inspect message to an instantiated object. An Inspector window with two panes—one on the right and one on the left—then

opens. The pane on the left displays the names of the instance variables, and the right pane shows their values.

Two fonts come with Smalltalk/V. The fonts differ mainly in size—one has 8 × 8 pixels and the other 8 × 14 pixels. Other features included in Smalltalk/V that were absent in Methods are a DOS shell, a garbage collector, and virtual memory management.

The Smalltalk/V manual is the *Tutorial and Programming Handbook*. In many respects it is remarkably clear and well written. Its main shortcoming is that, in spite of its thoroughness and readability, it is not a complete reference to the behavior of the Smalltalk/V system. I would like to see a companion reference guide that would go more deeply into the behavior and implementation of the garbage collector and virtual memory, for example.

Graphics

An interesting approach to graphics has been adopted in Smalltalk/V. The basic class that implements the graphics capability is the BitBlt class. which is named for the bit block transfer operation and is much as in Smalltalk-80. Together with its immediate subclasses Pen and CharacterScanner and the subclasses of Pen-Commander and Animation - BitBlt provides the basis for how Smalltalk/V creates bit-mapped displays. The block transfers occur between two Forms—a source Form and a destination Form. Forms, Points, and Rectangles constitute the main structures used in Smalltalk/V graphics. A mechanism called a clipping rectangle is used-again, much as in Smalltalk-80—to define the maximum size of the bit transfer. This clipping rectangle restricts the size of the rectangular array of bits that will constitute the destination Form.

To see how this works, first look at the section of the class hierarchy in question:

BitBlt

CharacterScanner
Pen
Animation
Commander

The CharacterScanner class has the

PVCS The Most Powerful & Flexible Source Code Revision & Version Control System.

The POLYTRON Version Control System (PVCS) allows programmers, project managers, librarians and system administrators to effectively control the proliferation of revisions and versions of source code in software systems and products. PVCS is a superb tool for programmers and programming teams. If you allow simultaneous changes to a module, PVCS can merge the changes into a single new revision. If changes conflict, the user is notified. Powerful capabilities include: Storage and retrieval of multiple revisions of text; Maintenance of a complete history of revisions to act as an "audit trail" to monitor the evolution of a software system; Maintenance of separate lines of development or "branching"; Levels of security to assure system integrity; An intelligent difference detector to minimize the amount of disk space required to store a new version. Requires DOS 2.0 or higher. Compatible with the IBM PC, XT, AT and other MS-DOS PCs. Maintains source code written in ANY language.

Only PVCS meets the needs of Independent Programmers and Corporations. Once you standardize on PVCS, the "Logfiles" used to track and monitor changes are interchangable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced PVCS.

Personal PVCS — Offers most of the power and flexibility of the Corporate PVCS, but excludes the features necessary for multiple-programmer projects. \$149

Corporate PVCS — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes "Branching" to effectively maintain code when programs evolve on multiple paths (e.g., new versions for different systems, or a new program based on an existing program). Single User License Price. \$395

Network PVCS — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project. 5-Station License \$1,000. Call (503) 645-1150 for pricing on Licenses for more than 5 Stations.

TO ORDER: VISA/MC 1-800-547-4000. Dept. No. 355.

Oregon and outside US call (503) 684-3000. Send Checks, P.O.s to: POLYTRON Corporation, 1815 NW 169th Place, Suite 2110, Dept. 355 Beaverton, OR 97006.



It is also interesting to see how Smalltalk/V handles windows. Here, the approach is very different from that used in Smalltalk-80 but is essentially the same as that used in Methods. The following segment of the class hierarchy comes into play:

Dispatcher
GraphDispatcher
PointDispatcher
ScreenDispatcher
ScrollDispatcher
FormEditor
ListSelector
TextEditor
PromptEditor
TopDispatcher
DispatchManager

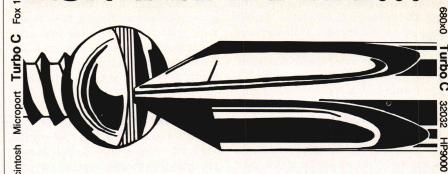
Browsing Drives

The Disk Browser is one of the more original and powerful facilities in the Smalltalk/V system. Its window is composed of four panes. In the upper-left pane is the directory hierarchy list, which shows all the directories on a disk. In the pane to the right of this is the file list, which displays the names of all the files in the selected directory. A large pane below these is the contents pane, which displays either the screen of directory information as it would appear after an MS-DOS dir command or the contents of a selected file. And just above this, there is a small pane called the directory order pane, which displays the way directory information is currently being sorted for display in the contents frame-such as by date, name, or size. With this facility you can create or remove directories and files as well as rename, copy, or print them.

With the command menus accessible from the contents pane, you can perform just about any file maintenance operation, including cut and

Microsoft Lattice Turbo C Aztec Computer Innovation Mark Williams Turbo C

ISN'T IT A PITY...



Everything Isn't As Accommodating As

c-tree / r

REPORT GENERATOR

FILE HANDLER

Performance and Portability

Novell

Turbo C

Xenix

Unix

O

O

For all the time you devote to developing your new programs, doesn't it make sense to insure they perform like lightning and can be ported with ease?

c-tree: Multi-Key ISAM Functions For Single User, Network, & Multi Tasking Systems

Based on the most advanced B+ Tree routines available today, **c-tree** gives you unmatched keyed file accessing performance and complete C Source Code. Thousands of professional C programmers are already enjoying **c-tree**'s royalty-free benefits, outstanding performance, and unparalleled portability.

Only FairCom provides single and multi-user capabilities in one source code package, including locking routines for Unix, Xenix, and DOS 3.1., for one low price! In addition, **c-tree** supports fixed and variable record length data files; fixed and variable length key values with key compression; multiple indices in a single index file; and automatic sharing of file descriptors.

r-tree: Multi-File Report Generator

r-tree builds on the power of c-tree to provide sophisticated, multi-line reports. Information spanning multiple files may be used for display purposes or to direct record selection. You can develop new reports or change existing reports without programming or recompiling and can use any text editor to

create or modify **r-tree** report scripts including the complete report layout. At your option, end users may even modify the report scripts you provide.

Unlimited Virtual Fields; Automatic File Traversal

r-tree report scripts can define any number of virtual fields based on complex computational expressions involving application defined data objects and other virtual fields. In addition, r-tree automatically computes values based on the MAX, MIN, SUM, FRQ, or AVG of values spread over multiple records. r-tree even lets you nest these computational functions, causing files from different logical levels to be automatically traversed.

Unlike other report generators, **r-tree** allows you to distribute executable code capable of producing new reports or changing existing reports without royalty payments, provided the code is tied to an application. Your complete source code also includes the report script interpreter and compiler.

How To Order

Put FairCom leadership in programmers utilities to work for you. Order **c-tree** today for \$395 or **r-tree** for \$295. (When ordered together, **r-tree** is only \$255). For VISA, MosterCard and C.O.D. orders, call 314/445-6833. For **c-tree** benchmark comparisons, write FairCom, 4006 West Broadway, Columbia, MO 65203.



Complete C Source Code & No Royalties!

Xenix is a registered trademark of Microsoft Corp.

Unix DOS Xenix Turbo C Novell VMS Macintosh Microport Turbo C Fox 10-NET

CIRCLE 93 ON READER SERVICE CARD

Computer Innovation Mark Williams Turbo C Unix D

3B2 Turbo C

11/73

C

ARTIFICIAL INTELLIGENCE (continued from page 131)

paste, copy, read, and install. The full set of commands available as displayed in the menu varies depending upon the size of the file. Normally, with files greater than 6,000 bytes, the contents pane displays only the first and last 2,000 bytes. In that event, you can use the read it command to read in the complete contents of a large file. Also available is the save as command, which allows a file to be saved under a different name. Finally, the install command allows source files to be compiled into the Smalltalk/V system.

One thing you have to keep track of is the size of the changes.log file. If it starts to get large, then there is a facility for compressing it. You must use this before the changes log gets too large and space on the disk runs low, or your image will die the death. The log facility is an essential one for those who can't resist taking advantage of the fact that Smalltalk is internally extensible to a large degree, as are Forth and LISP. While

modifying the internals to create an image of a new dialect of Smalltalk/ V, prior to getting your modified system debugged, it is bound to crash more than twice. The log file is insurance that you will never lose anything you've done for keeps, unless for some reason a crash clobbers this file. If necessary, you can even use it to restore the system image.

A Method to the Madness

At the very center of all this are the methods-the actual modular subroutines that do the message passing. There are two different types of methods-instance methods and class methods, which are analogous to the instance and class variables.

One important departure of Smalltalk/V from the Smalltalk-80 standard is the omission of the ClassDescription class. In Smalltalk-80, the class hierarchy starting with the Behavior class is organized like this:

Behavior

ClassDescription

Class

Metaclass

The arrangement in Smalltalk/V is the same except that ClassDescription is omitted. As a result, Smalltalk/ V does not support message categories-that is, the grouping of methods for a given class under various category names. In many cases, I have found it relatively easy to add missing Smalltalk-80 classes to Smalltalk/V. In this case an addition is not easy to make because, if ClassDescription is added as a subclass of Behavior, it becomes a peer class of Class and Metaclass rather than a superclass of them.

On the other hand, Smalltalk/V has several special classes that are not found in Smalltalk-80. Table 1, below, contains a list of those that are not simply machine-specific classes but have to do with the IBM PC implementation and user interface.

Example: A Small Inference Engine

To give you a better grasp of specific things that Smalltalk/V can do, I'll now discuss an example of a simple rule-based reasoning system that was provided by Digitalk. First, let's look at its overall structure.

There is a kind of application holder class called InferenceEngine that has three subclasses—Expert, Fact, and Rule-that do all the work. An instance of the class Expert is a working inference engine that can evaluate rulebases for particular subject areas or domains. In this case it is a tree expert. The inference engine is a simple forward-chaining one that accepts a set of facts and applies the rules to the facts it already knows to determine if the goal is true. It then allows you to ask it to explain how it

The Advanced Programmer's Editor That Doesn't Waste Your Time

EPS (e

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
 Great on-line help system
- Multiple windows, files
- Unlimited file size, line length
 Supports large displays
- Regular Expression search
- 30 day money-back guarantee
 Not copy protected

Only \$195

5740 Darlington Road Pittsburgh, PA 15217



for IBM PC/XT/AT's or compatibles

CursorManager Directory

DiskBrowser Dispatcher

DisplayString FixedSizeCollection

IndexedCollection

InfiniteForm

LinkedListStream MethodDictionary

StringBlt StringModel

SystemDictionary

Table 1: Classes unique to Smalltalk/V

HOT GRAPHICS PACKAGE FOR C PROGRAMS: \$39.95

verything you need to write dramatic graphics effects into your Eco-C88 C programs. Some of the features include:

- Support for EGA, CGA, and Z100
- Over 100 graphics and support functions, many of which are PLOT-10 compatible.
- Many low level support routines reside outside your small model code-data area
- Can write dots thru the BIOS (for compatibility) or to memory (for speed)
- Graphics function help from CED editor available
- World, pixel or turtle color graphics modes
- 47 standard fill patterns, 17 line dashing patterns, Hershey fonts, plus user defineable fill, dash and fonts
- Supports view areas, rotateable fonts, clipping, arbitrary fill areas, extensive error checking, examples, and user's manual.

A must for the graphics enthusiast and a bargain at only \$3095

*Requires Eco-C88 C Compiler.

NEW POP-UP WINDOWS FOR YOUR C PROGRAMS

This windowing library allows you to add pop up windows in your C programs quickly and easily. Use them for help windows, selection menus, error messages, special effects—anywhere you need an attention getter. Just some of the features include:

- CGA, EGA, and monochrome support
- Slow mode option for "flicker" displays
- Control any program that goes through the BIOS

- Use up to 255 windows
- No special window commands; use print f ()
- Resize and move windows
- Custom window titles and borders
- · Can be used with ANSI device driver
- Most of window's code-data lies outside small model limits
- Use any of the IBM text or block characters
- User's manual and examples

The Windowing Library requires an IBM PC compatible BIOS and the Eco-C88 C compiler.

ONLY \$29.95

HANDY LIBRARIAN MAKES LIFE EASIER.

Now you can combine your modules, functions, and subroutines into your own library for easy link commands. Fully compatible with ANY standard OBJ format files (not just Ecosoft's products). With the Ecosoft librarian, you can:

- · Add, delete, and extract from a library
- Get table of contents or index of a library
- Combine libraries, control library page size, use switches for combinations, process complex library requests, use wildcards, and do library directives from command files.
- Complete with user's manual
 A valuable addition for any programmer.

ONLY \$29.95

Orders only:

1-800-952-0472 Technical Information:

(317) 255-6476



THE FIRST PROFESSIONAL 'C'COMPILER FOR UNDER \$60.

A C compiler with many ANSI enhancements at an unbelievably low price. The Eco-C88 C compiler has:

- Prototyping (the new type-checking enhancement)
- Enum and void data types
- Structure passing and assignment
- All operators and data types
- A standard library with more than 200 functions (many of which are System V compatible for greater code portability)
- CC and mini-make that all but automates the compile process
- 8087 support (we sense the 8087 at runtime
 no dual libraries)
- ASM or OBJ output for use with MSDOS linker
- Tiered error messages enable-disable lintlike error checking
- · Fast compiles and executing code
- Expanded user's manual
- CED full-screen program editor

Everything you need at the unbelievable price of \$59.95.

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later.

Ecosoft Inc. 6413 N. College Ave. Indianapolis, IN 46220

ECO-C88 COMPILER Support Products Support Products Fig. 30. Fi

ORDER FORM CLIP &	MAIL TO: Ecosoft I	nc., 6413 N. Colle	ge Ave., Indianapolis,	IN 46220
ITEM	PRICE	QTY	Т	OTAL
Flexi-Graph Graphics	\$39.95			
Window Library	\$29.95			
Eco-Lib Librarian	\$29.95			
Eco-C88 C Compiler CED	\$59.95			
		SHIF	PPING	
	TOTAL (IN	D. RES. ADD 5%	TAX)	
PAYMENT:	□ VISA	□ мс	□ AE	□ СНЕСК
CARD #			EXPIR DATE	
NAME				
ADDRESS				
CITY			STATE	
ZIP	PHONE			
	CIRCLE 89 OF	N READER SERV	ICE CARD	

ECOSOFT

ARTIFICIAL INTELLIGENCE (continued from page 132)

arrived at its result. The main operation of the inference engine is to apply rules to test a result. It sends a message to the rules collection to test the facts and then applies the eval: method to see if the goal to be tested matches. If so, it returns true; if not, it returns false.

The action method fires the conclusion of a rule and in doing so adds a new fact to the facts collection. The way you use this little demo expert system is by entering everything it needs directly in a text area and then causing the interpreter to evaluate it by selecting DO IT from the dropdown menu. This does the following things: First, it creates an instance of the Expert class called by the global variable name Tree; second, it initializes an instance of the Fact class; and third, it submits various facts about a tree for evaluation by adding them to the Fact collection. This sends the message to the Tree expert to prove the goal that, given the previously entered facts, for example, the tree is of the cyprus family.

Several things should be noted about this inference engine demo, which was not intended to be anything more than a simple toy demonstration. First, it does not use a rule syntax other than Smalltalk itself, so no parser is needed. This allows it to run quickly but makes the rules less readable-a familiar trade-off. Another point is that it lacks the ability to read in the fact and rule collections directly from a file—a facility that could be added without too much difficulty. There is also no real user interface other than the explain method. In particular, there is no facility for posing questions to users about values that the system cannot find otherwise. The point is that, although this is a toy system, it points to what a full system might be. Most of these features are not too difficult to add, once you grasp the basics of how to implement an inference engine in Smalltalk.

The advantages of implementing a full expert system shell in Smalltalk/ V are quite easy to see. First, you get the lush, easy-to-understand user interface practically for free. It would not be a major development project to use the Smalltalk/V menu and windowing facilities to build a superior expert system consultation environment. Another important plus is the multiple instance aspect of Smalltalk. You can have as many Experts, like the Tree expert, initialized as necessary, each with a separate ruleset. Smalltalk/V could then be programmed for one Expert to pass a message to another for a goal to be tested. Also, more flexible inference methods could be implemented for backward chaining and combining both forward and backward chaining. Finally, a parser could be written that could accept a more "friendly" rule syntax and compile it into the Smalltalk format as used here for running finished knowledge bases.

Conclusions

Smalltalk/V, Version 1.2, is a remarkable accomplishment and a very easy-to-understand environment for newcomers to Smalltalk to get acquainted with the language. Its performance is surprisingly fast, considering all the things going on.

I should also mention that the Goodies Extension Kit disk, which is now available as an option for Smalltalk/V, contains an implementation of multiprocessing that could be important if discrete simulation is what you're after in a Smalltalk environment. The simulation examples in the Xerox Smalltalk series presume the multiprocessing capability of Smalltalk-80. Now, with the multiprocessing classes provided on the optional Goodies disk, this should not be a problem.

DDJ

Vote for your favorite feature/article. Circle Reader Service No. 7.

Brand New From Peter Norton A PROGRAMMER'S EDITOR

only

Direct from the man who gave you The Norton Utilities, Inside the IBM PC. and the Peter Norton

Programmer's Guide.

that's lightning fast with the hot features programmers need

"This is the programmer's editor that I wished I'd had when I wrote my Norton Utilities. You can program your way to glory with The Norton Editor'

Peter Nortan

Easily customized, and saved Split-screen editing A wonderful condensed/outline display Great for assembler, Pascal and C

Peter Norton Computing, Inc., 2210 Wilshire Boulevard, Santa Monica, CA 90403, 213-453-2361. Visa, Mastercard and phone orders welcome.

The Norton Editor™ is a trademark of Peter Norton Computing, Inc. © 1986 Peter Norton Computing

CIRCLE 243 ON READER SERVICE CARD



Smalltalk/V Digitalk Inc. 5200 Century Blvd. Los Angeles, CA 90045 (213) 645-1082 Reader Service No. 127

MICROWAY ACCELERATES YOUR PC!

FastCACHE-286™

Runs your PC Faster than an AT! Runs the 80286 at 9 or 12 MHz and the 80287 at 8, 9 or 12 MHz, Includes 8 kbytes of 55ns CACHE.



Compatible with IBM PC, XT, Leading Edge Model D, Compaq, and Turbo motherboards. Includes 8088 Reboot Switch, DCache, Print Spooler and Diagnostics From \$399

8087 SOFTWARE

	Z. 10 Y
IBM BASIC COMPILER	\$465
MICROSOFT QUICK BASIC	\$79
87BASIC COMPILER PATCH	\$150
87BASIC/INLÍNE	\$200
IBM MACRO ASSEMBLER	\$155
MS MACRO ASSEMBLER	\$99
87MACRO/DEBUG	\$199
MICROSOFT FORTRAN V4	
RM FORTRAN	\$399
LAHEY FORTRAN F77L	
MS or LATTICE C	. CALL
STSC APL★PLUS/PC	\$450
STSC STATGRAPHICS	\$675
SPSS/PC+	\$695
87SFL Scientific Functions	\$250
87FFT	\$200
OBJ → ASM	
PHOENIX PRODUCTS	
1110211111111000010 11111111	

AT8TM

Turns your AT into a high speed, multi-user Xenix business system!

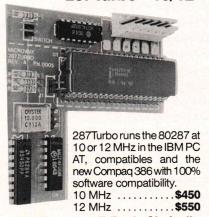


8 port, intelligent serial controller with 3% response degradation. Includes 8 MHz 80186 with built in DMA \$1299

LOTUS/INTEL EMS **SPECIFICATION BOARDS**

MegaPage™ The only EMS board which comes populated with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache and EMS drivers. For the IBM PC, XT and compatibles . . . \$549 MegaPage with ØK\$149 MegaPage with 2 megabytes of HMOS RAM\$419 MegaPage AT/ECC™ EMS card for the PC AT and compatibles includes Error Correction Circuitry. With ECC, 11 RAM chips cover 256K so the user never encounters RAM errors. With 1 megabyte CMOS RAM\$699 INTEL, JRAM, or Maynard CALL INTEL INBOARD 386 ØK\$1250

287 Turbo™-10/12



PC Magazine "Editor's Choice"

NUMBER SMASHER/ECMT



12 MHz 8086/8087 **Accelerator** Plus

A Megabyte for DOS!

For the IBM PC, XT and compatibles PC Magazine "Editor's Choice"

8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and installation instructions.

8087 5 MHz\$99
For the IBM PC, XT and compatibles
8087-2 8 MHz \$154

0007-2 0 WILLS
For Wang, AT&T, DeskPro, NEC, Leading Edge
80287-3 5 MHz\$159
80287-6 6 MHz \$179

For 8 MHz AT and compatibles

80287-8 8 MHz**\$259** For the 8 MHz 80286 accelerator cards

80287-10 10 MHz\$395 80387-16 16 MHz\$495 PC-PAL™ Programmer\$395

64K 150ns\$15 256K 150ns\$36

Call for great prices on V20 & V30

MICROWAY SOFTWARE FOR LOTUS 1-2-3™

PowerDialer® Add-In for Lotus 1-2-3 Release 2. Automated telephone dialing from within 1-2-3. Adds least cost routing, automatic carrier selection and automated phone book worksheet. Builds customized dialing applications. Can be used

FASTBREAK™ employs the 8087 to increase the speed of Lotus 1-2-3™ Version 1A or 1A*. Users are reporting speed ups of between 3 and 36 to 1. When run with our NUMBER SMASHER accelerator card, recalculation speed ups of 10 to 30 are being reported\$79

HOTLINK™ adds easy linking of spreadsheets to Lotus 1-2-3 Version 1A . . . \$99

287 TURBO-PLUS™ Speeds up your AT

Adjustable 80286 Clock 6-12 MHz 10 MHz 80287 Clock Plus Full Hardware Reset ... Optional 80286-10\$175



287TURBO-PLUS

With 80287 10 MHz\$549 With 80287 12 MHz\$629

CALL (617) 746-7341 FOR OUR COMPLETE CATALOG

P.O. Box 79 Kingston, Mass. 02364 USA (617) 746-7341

The World Leader in 8087 Support!

MicroWay Europe 32 High Street **Kingston-Upon-Thames** Surrey England KT1 1HL Telephone: 01-541-5466

C CODE FOR THE PC

source code, of course

C Source Code	
FSP (screen manager)	\$400
Bar Code Generator (Codabar, 3 of 9, and other popular codes; price is per code)	\$300
GraphiC 4.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	\$275
Vitamin C (MacWindows)	\$200
Essential C Utility Library (400 useful C functions)	\$160
Essential Communications Library (C functions for RS-232-based communication systems)	\$160
Panache C Program Generator (screen-based database management programs)	\$150
PC/IP (CMU/MIT TCP/IP implementation for PCs)	\$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	\$100
The Profiler (program execution profile tool)	
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	\$00
CBTree (B+tree ISAM driver, multiple variable-length keys)	. \$90
ME (programmer's editor with C-like macro language by Magma Software)	. \$00 \$75
Wendin PCNX Operating System Shell	
Wendin PCVMS Operating System Shell	. \$75
Wendin Operating System Construction Kit	. \$75
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	. \$70
EZ_ASM (assembly language macros bridging C and MASM)	. \$60
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	
Make (macros, all languages, built-in rules)	
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	
Coder's Prolog (inference engine for use with C programs)	. \$45
PC/MPX (light-weight process manager; includes preemption and cooroutine packages)	. \$45
Biggerstaff's System Tools (multi-tasking window manager kit)	. \$40
TELE Kernel (Ken Berry's multi-tasking kernel)	. \$30
TELE Windows (Ken Berry's window package)	. \$30
Clisp (Lisp interpreter with extensive internals documentation)	. \$30
Translate Rules to C (YACC-like function generator for rule-based systems)	. \$30
6-Pack of Editors (six public domain editors for use, study & hacking)	. \$30
ICON (string and list processing language, Version 6 and update)	. \$25
LEX (lexical analyzer generator)	. \$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	. \$25
C Compiler Torture Test (checks a C compiler against K & R)	. \$20
PKG (task-to-task protocol package)	. \$20
A68 (68000 cross-assembler)	. \$20
Small-C (C subset compiler for 8080 and 8088)	. \$20
tiny-c (C subsubset interpreter including the tiny-c shell)	. \$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	. \$20
List-Pac (C functions for lists, stacks, and queues)	. \$20
XLT Macro Processor (general purpose text translator)	. \$15
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	. \$15
(I work of Friends, Start, Lands, La	. 415
Data	
Protein Sequences (roughly 4,000 protein sequences with similarity search program)	. \$60
Webster's Second Dictionary (234,932 words)	. \$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	. \$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	. \$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TrX or bitmap format)	. \$30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	\$15
U. S. Map (15,701 points of state boundaries)	. \$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas USA 78750-3409
(512) 258-0785

For Free Info ...

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



Start Here

Smart buyers start with DDJ's free information card, a shopping center filled with information about the products and services advertised in this very issue: everything from software and systems to peripherals and professional support services.

And smart buyers can use this free information card to quickly and easily gather a comprehensive file of facts, figures and product specs to sort out competing claims. Using *DDJ*'s free information card can prevent you from making the wrong, costly buying decision.

Be a smart shopper. Complete and mail this postage paid card today!

SS REPLY MAI

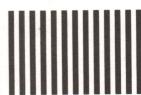
FIRST CLASS PERMIT #217, CLINTON, IOWA
POSTAGE WILL BE PAID BY ADDRESSEE

TO THE PAID BY ADDRESSEE

P.O. Box 2157 Clinton, Iowa 52735-2157

Take a Reader Service Card with You

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



It's Easy as ...

Circle the appropriate free information numbers, referring to the advertiser index for more information.

2 Fill in your name and address.

Mail today—postage is absolutely free. You'll receive the product information you need directly from the manufacturers, thanks to DDJ.



POSTAGE WILL BE PAID BY ADDRESSEE



P.O. Box 2157 Clinton, Iowa 52735-2157

Dr. Dobb's Journal of Software Tools

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 22 21 22 22 22 22 22 22 22 22 22	26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47	51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 70 71 72	76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97	101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122	126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146	151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 170 171 172	176 177 178 179 180 181 182 183 184 185 186 187 198 199 191 192 193 194 195 196	202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222	226 227 228 229 230 231 232 233 234 235 236 237 238 240 241 242 243 244 245 244 247	251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 270 271 272	276 277 278 279 280 281 282 283 284 285 286 287 288 290 291 292 293 294 295 296 297	301 302 303 304 305 306 307 308 309 311 312 313 314 315 316 317 318 320 321 322	326 327 328 329 330 331 332 333 334 335 336 337 338 340 341 342 343 344 345 346 347	351 352 353 354 355 356 357 358 369 361 362 363 364 365 366 367 368 369 370 371 372	376 377 378 379 380 381 382 383 384 385 386 390 391 392 393 394 395 396 397	
21	46	71	96	121	146	171	196 197 198 199	221	246	271	296	321	346	371	396	

September '87: Use before December 31, 1987

Name	
Title	
Company	Phone
Address	
City/State/Zip	

Dr. Dobb's Journal of Software Tools FOR THE PROFESSIONAL PROGRAMMER

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 22 23 24	26 27 28 29 30 31 32 33 34 35 36 37 38 40 44 44 45 46 47 48 49	51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 71 72 73 74	76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99	101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124	126 127 128 129 130 131 132 133 135 136 137 138 139 140 141 142 143 144 145 146 147	151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 170 171 172 173 174	177 178 179 180 181 182 183 184 185 186 187 198 190 191 192 193 194 195 196 197 198	201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 222 223	238 239 240 241 242 243 244 245 246 247 248	252 253 254 255 256 257 258 259 261 262 263 264 265 266 267 268 269 271 272 273	277 278 279 280 281 282 283 284 286 287 288 290 291 292 293 294 295 296 297 298	302 303 304 305 306 307 308 309 311 312 313 314 315 316 317 318 319 321 322 323	326 327 328 329 330 331 332 333 335 336 337 338 340 342 343 344 345 346 347 348	351 352 353 354 355 356 357 358 360 361 362 363 364 365 367 368 369 370 371 372 373	376 377 378 379 380 381 382 383 384 385 386 387 390 391 392 393 394 395 396 397 398	
			98 99 100	123 124 125	148 149 150	173 174 175	198 199 200	223 224 225	248 249 250	273 274 275	298 299 300	323 324 325	348 349 350	373 374 375	398 399 400	

September '87: Use before December 31, 1987

Name	
Title	
Company	Phone
Address	THOME
City/State/Zip	

For Free Info ...

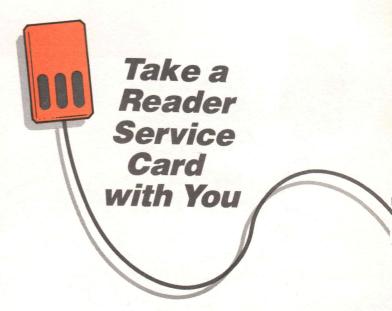
Start Here

Smart buyers start with DDJ's free information card, a shopping center filled with information about the products and projects advertised in this year issue; even thing from software.

services advertised in this very issue: everything from software and systems to peripherals and professional support services.

And smart buyers can use this free information card to quickly and easily gather a comprehensive file of facts, figures and product specs to sort out competing claims. Using *DDJ*'s free information card can prevent you from making the wrong, costly buying decision.

Be a smart shopper. Complete and mail this postage paid card today!



It's Easy as ...

Circle the appropriate free information numbers, referring to the advertiser index for more information.

2. Fill in your name and address.

Mail today—postage is absolutely free. You'll receive the product information you need directly from the manufacturers, thanks to DDJ.



The Advertiser Index

Advertiser Name	Page No.	Circle No.
Al Architects	31	265
Aker Corporation	81	369
Aldebaran Laboratories		350
Alsys	46-47	273
Aptech Systems, Inc.	69	*
Austin Code Works		250
Blaise Computing		159
Borland International		161
Boston Software Works (The)		384
Bryte Computer	65	387
Burton Systems Software		212
C Tools		
C Users Group		181
Cadre Technologies, Inc		261
Cobalt Blue		370
Compu View		122
Creative Programming		474
Crosstalk Communications		171
Custom Software Systems		268
DDJ Subscriptions	80	000
Datalight	9	203
Desktop A.I.	64	258
Digitalk	146	127
Ecosoft, Inc.		89
Enertec		346
Essential Software		138
Fair-Com		93
Flexus		189
Galacticomm		362
Galacticomm		342
GenSoft Systems	93	166
Gimpel Software		97
Greenleaf Software		351
Guidelines Software	1/7	274
Hauppauge Computer Works	1/0	376
Intel Corporation		179
JYACC		146
Logic Process Corporation	102	169
Logic Process Corporation		226
Logitech, Inc.		257
Lugaru Software Ltd.	132	135
M Street Software	66	276
MS-DOS Tools		*
M&T Catalog of Books & Software Tools	PB	*
Machine Independent Software Corp		294
Magma Systems		313
Manx Software Systems	7	108
Mark Williams Company		102
Meridian Software Systems	41	397
Metagraphics Software Corporation	83	392
MetaWare Incorporated	78	95
Micro Way	135	300
Microcompatibles	78	286
Microprocessors Unlimited	87	105
Microsoft	32a	380
Microsoft Press	115	125
Microsoft Press		126
MMC AD Systems	119	192
Mortice Kern Systems, Inc	75	249
Nanosoft Associates	101	309
Nantucket Corporation	27	220
New BASICs	62	*
Norton Utilities (The)	134	243
Norton Utilities (The)		7 87
Oakland Group, Inc. (The)	110	227

Advertiser Name	Page No.	Circle No.
Oasys		254
Oregon Software		357
Palo Alto Shipping	.96	76
Periscope Co., Inc		214
Pharlap		343
PIM Publications	.79	136
Pioneering Controls Technologies, Inc	.53	191
PMI	.89	239
Polytron Corporation		283
Prentice Hall		*
Programmer's Connection	.149-151	129
Programmer's Connection	.23	98
Programmer's Shop (The)	.143	133
Quantum Computing	.71	144
Quarterdeck Office Supplies	.10	284
Raima Corporation		*
Rainbow Technologies		255
SAS Institute	.141	*
Santa Rita Software	.15	353
Scantel Systems Limited	.96	391
Scientific Endeavors	.66	210
Secom Information Products Co	.76	394
Seidl Computer Engineering	.55	114
Semi-Disk Systems		85
Sharpe Systems Corporation	.34	176
SLR Systems	.77	78
Softcraft Inc		113
Softfocus	.79	259
Software Garden Inc		314
Software Security, Inc		170
Solution Systems		142
Solution Systems		152
Texas Instruments		*
Texas Instruments		*
Tool Makers (The)		319
True Basic		344
TSF (The Software Family)		230
Turbo Tech Report		119
Unipress Software		77
Vermont Creative Software	.43	157
Wallsoft		90
Wendin		112
Whitewater Group (The)		282
Wordcraft		163
Zark Incorporated		164
ZyLAB		329
_,		

*This advertiser prefers to be contacted directly; see ad for phone number.

Advertising Sales Offices

Midwest

Charles Shively (415) 366-3600
Northern California/Northwest
Lisa Boudreau (415) 366-3600
Northeast
Cynthia Zuck (718) 499-9333
Martha Brandt (415) 366-3600
Southern California/AZ/NM/TX
Michael Wiener (415) 366-3600
Director of Marketing and Advertising
Ferris Ferdon (415) 366-3600

(continued from page 14)

to...Oh my God—he did forget!). Stan Kelly-Bootle 25 Parkwood Mill Valley, CA 94941

Dear DDJ,

Brian Anderson states that "often the only control structures available in assembly language are the conditional and unconditional jump and the call to subroutine." The 68000 and most other modern processors have looping structures, generally provided for use by high-level language compilers. Therefore, I offer my Example 2, below, as a replacement for his. Also, notice that this code does not get stuck in an endless loop, unlike its predecessor.

I would like to thank Mr. Anderson for his interesting and otherwise excellent article.

Matthew Siegel 192 Belvedere St., #9 San Rafael, CA 94901-4748

Dear DDJ,

I guess I still have a bit of egg on my face—it seems that I can't program my way out of an infinite loop (at least when it comes to 68000 assembly language).

The 68000 example that I cited in my Viewpoint won't work because I forgot to increment the index variable inside the loop. My wife thought that, because this mistake supports my point (sort of), I should claim that I planted the error just to see if anyone would catch it.

Please let me assure you, the mistake was an honest one. My apologies to *DDJ* readers (especially 68000 hackers).

Brian Anderson 5105 Lorraine Ave. Burnaby, BC Canada V5G 2S3

Math and Programming

Dear DDJ,

I feel I must take issue with Allen Holub concerning his Viewpoint in the April 1987 issue of *DDJ*.

To begin with, from personal experience, I can draw a strong correlation between advanced mathematics and the art of programming. There is a definite parallel between juggling a large system of algebraic equations in your head and trying to maintain the purpose and use of many interacting variables in a computer program. Obviously implementations of some of the purer algorithms in computer science such as queueing theory, graphs and trees, sorting algorithms, vector math, automata theory, frequency analysis, formal logic, cubic splines, compression algorithms, minimax and Bayes decision theory, branch and bound problems, probabilistic and statistical concepts, game theory, and all aspects of operations research require a deep understanding of algebraic notation, linear algebra, and many other forms of abstract representation.

Allen implies that mathematics is nothing more than applying a set of memorized rules to a problem. I beg to differ if he feels that solving a third-order differential equation or a partial derivative is not a creative problem solving process. Those "memorized rules" are less rules and more approaches and guidelines

with which to tackle the problem as stated. I find solving calculus and differential equations not at all unlike trying to come up with my own algorithms for a computer simulation problem. I feel that those students who could not tackle and solve advanced mathematics would also have great difficulty in implementing their own fresh approaches to new and yet unsolved computer science problems.

In agreement with Allen, I would state that if a student's goal is to become an applications programmer who is never responsible for an original algorithm but who instead simply implements code and algorithms found in books, then by all means forego anything beyond Algebra I. Allen covers himself by stating that math is not a prerequisite for programming as it is for engineering. What does he think the name computer scientist means anyway? And what kind of jobs do you think computer scientists have? They work on developing new approaches in medicine, vision processing, knowledge

```
68000 assembly language FOR loop
  The first element to clear is FIRST, the last element LAST.
  Character data begins at DATA.
* REGISTER USAGE
  A0 points to the current element of the array.
  D0 = one less than the number of elements left to clear.
FIRST DS.W 1
LAST DS.W 1
DATA DS.B 500
CLEAR LEA DATA, AO
                           ; A0 points to character data
   MOVE.W FIRST, DO
                           ; D0 = first element to clear
   LEA 0(A0, D0.W), A0
                           ; A0 points to first element to clear
   MOVE.W LAST, DO
                           ; D0 = number of elements to clear
   SUB.W FIRST, DO
                               (less 1, for DBcc loop use)
LOOP CLR.B (A0)+
                           ; clear an element and advance
   DBF DO, LOOP
                           ; repeat
```

Example 2: Loop structure correction for Brian Anderson's 68000 assembly code

THE 150% SOLUTION FOR SUPERIOR DATABASE DEVELOPMENT AT 62% OFF.

PHACT-manager™ gives MS-DOS™ programmers the ISAM they need. Plus a Report Writer, Query Language and full C source code.

- Efficient B + Tree access method.
- Unlimited number of keys and variable length records.
- Security: Password protection, shared/exclusive use.
- Runs on networks.
- Sequel-like query language for interactive or batch query/update.
- Report Writer: Perform "joins," create and use variables, sort, format and more.
- Versions for all popular C compilers.
- Thousands of licenses sold.

To order or get more information, call us at 1-800-222-0550 (outside NJ) or 1-201-985-8000 now.

MasterCard/Visa accepted.

Only \$249 complete! Full C source code.
No royalties!

UniPress Software

UniPress Software, Inc. 2025 Lincoln Highway Edison, NJ 08817

MS-DOS is a trademark of Microsoft. PHACT-manager is a trademark of PHACT Associates.

representation, flight simulation, computer games, the defense industry, and all forms of real-world simulation and control. One hardly needs a computer scientist to write a dBASE III application, yet the fields mentioned above seek out and demand a computer scientist with heavy mathematics background. This group of programmers does not make up a small specialized percentage but instead represents what a computer science degree is all about. Would you hire an MIT computer science graduate who had no more than high-school geometry to his or her credit? Neither would I.

If Allen rewrote the entire View-point and replaced each occurence of "computer scientist" with "data processing programmer," it would be a valid and important commentary. A dBASE III, 4GL, or COBOL programmer has little need for calculus, but for those of us breaking new frontiers in image processing, problem solving, and other areas of computer science, the need for a strong background in mathematics and formal symbolic representation is clear.

John W. Ratcliff 2510 LaCaracas St. Louis, MO 63114

Allen Holub replies:

There are several ways to learn how to manage large systems, and I still believe that mathematics is among the poorest of these, primarily because of the amount of background that you need just to get started. It takes a year and a half of college-level math to get to the point where you can start solving third-order differential equations, but most people (hopefully) know the rudiments of English composition before they get out of high school. Moreover, most holders of CS bachelors degrees don't know how to solve differential equations at all, for the simple reason that the courses aren't usually required. Mr. Ratcliff is correct in saying that higher mathematics can be useful to a programmer. Mathematics at the undergraduate level is not. It seems to me a waste of time to acquire the skills necessary to understand differential equations if all you really need to do is understand how to approach complex systems. It's not that mathematics won't get you there eventually but that there are better and faster ways to acquire the same skills, notably English composition.

Mr. Ratcliff's also contends that a strong background in "formal symbolic representation" is necessary. Again, I disagree. An ability to work with abstraction is, of course, necessary, but remember that logic started out life as a branch of philosophy, not of mathematics. Much of the basic work in compiler theory was done by linguists (such as Noam Chomsky at MIT), not by mathematicians. More often than not, a "formal symbolic representation" serves to obfuscate, rather than clarify. Good examples of this obfuscation can be found in the "dragon" book (Aho, Sethi, and Ulman), written by mathematicians and used in most compiler-design classes (even, I'm reluctant to admit, in the one that I teach). Aho spends pages inventing "formal symbolic representations" and then spends five lines actually explaining something useful. I'd rather spend half an hour reading a clear description of a process in English than spend the same half hour trying to decipher five lines of formal symbols. More to the point, I've found that students who find Aho incomprehensible have no trouble at all understanding the concepts, once these concepts are presented to them in a clear way that doesn't use Aho's formal symbolic notation.

Call for Recipes

Dear DDJ,

In his boffo review of our book, *Numerical Recipes: The Art of Scientific Computing* (May 1987), Joe Marasco mentions that we want to hear from readers who would like to see a C version of the book and source-code "recipes."

Actually, preparation of *Numerical Recipes in C* is well underway. In fact, we would like to hear from *DDJ* readers with an interest in beta-testing the C recipes. We'll happily send free beta diskettes to the first hundred people who respond, plus an additional number to readers who

can describe (in a sentence or two) their strong qualifications.

We view Numerical Recipes as a cooperative project between authors and users. It's time to get good, cheap, open-source numerical software into the C world, before the vendors of proprietary object-only libraries get too established, as was historically the case in the FORTRAN world (much to the grief of FORTRAN programmers).

William Press Numerical Recipes Software P.O. Box 243 Cambridge, MA 02238

Correction

Dear DDJ,

Thank you for covering PC-MOS/386 from The Software Link in two articles in your July 1987 issue. Unfortunately, some of the information in each article is incorrect. I'd like to take this opportunity to bring it to your attention.

In the article "Developing 80386 Applications...Today," the prices given for PC-MOS/386 are wrong. The single-user/multitasking version is \$195, the five-user version is \$595, and the twenty-five-user version is \$995.

In the 16-Bit Software Toolbox column The Software Link is erroneously referred to on page 106 as "a company previously known for its copyprotection schemes." The Software Link has never been involved in any type of copy-protection—we develop and manufacture multiuser/multitasking software.

Thank you for the opportunity to point out these errors of fact. Again, we appreciate your editorial coverage. Our programming staff considers *Dr. Dobb's* a valuable resource and we are grateful for your coverage of our products.

Colleen G. Goidel The Software Link 3577 Parkway Ln. Atlanta, GA 30092

DDJ

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a firstrate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- Generation of reentrant object code. Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- Optimization of the generated code. We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- Generated code executable in both 24-bit and 31-bit addressing modes. You can run compiled programs above the 16 megabyte line in MVS/XA.
- Generated code identical for OS and CMS operating systems. You can move modules between MVS and CMS without even recompiling.
- Complete libraries. We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library

routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix*-style I/O access method.

■ Built-in functions. Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix linkeditor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

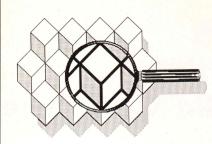
Complete and mail the coupon today. Because we've got the development tool for your tomorrow.

JAJ.

SAS Institute Inc. SAS Circle, Box 8000 Cary, NC 27511-8000 Telephone (919) 467-8000 x 7000

□ the C compiler for MVS□ the C compiler for CMS□ the cross-compiler with	S software developers	
todayso I'll be	ready for tomorrow	v.
Please complete or attach	your business card.	
Name		
Title		
Company		
Address		
City	State	ZIP
Telephone		

OF INTEREST



PS/2 Add Ons

Alloy Computer Products has announced three products for the PS/2 line: an internal tape drive and two adapters to connect its external tape drives and other products to the PS/2 machines. Reader Service No. 17. Alloy Computer Products Inc. 100 Pennsylvania Ave. Framingham, MA 01701 (617) 875-6100

Rodime has announced a hard disk on a card for the PS/2 Model 30, which it claims is "the only way for Model 30 users to get more than 20 megabytes of internal storage." The suggested retail price is \$1,495. Reader Service No. 18. Rodime Inc.

Peripheral Systems Division 29525 Chagrin Blvd., Ste. 214 Pepper Pike, OH 44122 (216) 765-8414

CMS Enhancements has exhibited external hard-disk subsystems for the PS/2 Model 30 as well as for the entire PS/2 line and for the Macintosh SE. CMS also has the first 5¹/₄-inch floppy drive for the PS/2, which should help all those early adopters move their software over to the PS/2 hardware. Reader Service No. 19. CMS Enhancements Inc. 1372 Valencia Ave. Tustin, CA 92680 (714) 259-9555

Kodak's diskette subsidiary, **Verbatim**, has announced a 2-megabyte 3.5-inch diskette (formatted capacity 1.44 megabytes). Reader Service No. 20. Verbatim Corp.
1200 W.T. Harris Blvd.
Charlotte, NC 28213
(704) 547-6500

Development Software

Sterling Castle, a new publisher of PC software, has introduced the Blackstar C function library designed to support the new ANSI standard and Microsoft, Version 3.0/4.0, and Lattice 3.0 C Compilers. Reader Service No. 21.

Sterling Castle Software 702 Washington St., Ste. 174 Marina del Rey, CA 90292 (213) 206-3020

The macro processing program SmartKey is evolving closer to a programming language as of its new version, 5.2, which adds context sensitivity and conditional processing. Programmer **Nick Hammond** wrote SmartKey back in 1979, and it is the original macro processor. It costs \$69.95. Reader Service No. 22.

Software Research Technologies Inc. 2130 South Vermont Ave. Los Angeles, CA 90007

(213) 737-7663

Alan Weiner has taken macro generation a step further. He has developed a memory-resident programming language, which he calls the Weiner Shell and which generates memory-resident programs. The program is written in assembly language, takes up less than 50K, and supports the LIM expanded memory spec, so any program written with it has access to up to 8 megabytes of LIM memory as well. It supports DOS interrupts, user I/O, arrays, and floating-point math. The idea of writing memory-resident programs on the fly that play around with DOS interrupts is scary, but Alan claims the product itself is robust. Its price is \$199. Reader Service No. 23. **Gryphon Microproducts**

P.O. Box 6543 Silver Spring, MD 20906 (301) 384-6868

Caro Research has a C code generator, developed by an outfit called Chancelogic PLC, called Pro-C that supports various ISAM file handlers; produces stand-alone C programs rapidly; and requires no end-user run-time system, royalties, or license. Reader Service No. 24. Caro Research Associates

Tampa, FL 33605 (813) 248-0852

202 South 22nd St.

Books and Stuff

We don't know why the company is called **Rabbit**, but we are compelled by this nomenclature to tell you that Rabbit has announced publication of its *Portable C and Unix Programming*, a 240-page reference guide for programmers writing applications in C or in the Unix environment. The book is available from Prentice-Hall for \$21.95. Reader Service No. 25. Rabbit Software Corp. Great Valley Corporate Center Seven Great Valley Parkway East Malvern, PA 19355 (215) 647-0440

AT&T also wants to educate you. It has announced a series of videotapes on Unix System V, Release 3. The tapes of possible interest to *DDJ* readers cover C and command shell programming and can be leased for \$300–375 a month or purchased for more. Reader Service No. 26.

AT&T Videotape Library (800) 247-1212

Windows Applications

Palantir Software has a shelf full of Windows applications, including word processing, spelling checking, scheduling, report generation, spreadsheeting, drawing, graphics scanning, and communications. Reader Service No. 27.
Palantir Software

12777 Jones Rd., Ste. 100 Houston, TX 77070 (713) 955-8880

Among the announced Windows applications at Comdex were two from **hDC**: an applications organizer called ClickStart, and EGA-16, a driver that doubles the number of displayable colors under Windows. The idea behind ClickStart seems to be that corporate users of Windows will need password protection of applications, turnkey menu selections, and restricted access to DOS functions. The vision of a traditional DP/MIS user interface on top of a point-and-shoot graphical user interface on top of the

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

RTC PLUS by Cobalt Blue. Translate FORTRAN 77 and RATFOR to Cexcept F77 I/O, FORTRAN character, and complex expressions. Some DEC F77 extensions. Library C Source. MS \$ 279

AI-Expert System Dev't

Arity Combination Package	PC	\$ 979
System - use with C	MS	\$ 229
SOL Dev't Package	MS	\$ 229
Auto-Intelligence	PC	\$ 739
CxPERT - shell for C	MS	
Experteach - Powerful, samples	PC	\$ 339
Exsys	PC	\$ 309
Runtime System	PC	\$ 469
Insight 2+	MS	\$ 379
Intelligence/Compiler	PC	\$ 739
T.I.: PC Easy	PC	\$ 435
Personal Consultant Plus	PC	\$ 2589
Personal Consultant Runtime	PC	\$ 85
Turbo Expert-Startup(400 rules)	PC	\$ 129
Corporate (4000 rules)	PC	\$ 359

Al-Lisp

Microsoft MuLisp 85	MS	\$	159
PC Scheme LISP - by TI	PC	\$	85
Star Sapphire	MS	\$	459
TransLISP - learn fast	MS	\$	79
TransLISP PLUS			
Optional Unlimited Runtime		\$	139
PLUS for MSDOS			169
Others: IQ LISP (\$239), IQC L	ISP (S	526	(9)

AI-Prolog

APT - Active Prolog Tutor - buil	d	
applications interactively	PC	\$ 49
ARITY Prolog - full, 4 Meg		
Interpreter - debug, C, ASM	PC	\$ 229
COMPILER/Interpreter-EXE	PC	\$ 569
Standard Prolog	MS	\$ 77
	1AC	\$ 269
MicroProlog - Prof. Entry Level	MS	\$ 85
MicroProlog Prof. Comp./Interp.	MS	\$ 439
MPROLOG P550	PC	\$ 175
Prolog-86 - Learn Fast	MS	\$ 89
Prolog-86 Plus - Develop	MS	\$ 199
TURBO PROLOG by Borland	PC	\$ 69

Basic

BAS_C - economy	MS	\$ 179
BAS_PAS - economy	MS	\$ 135
Basic Development System	PC	\$ 105
Basic Development Tools	PC	\$ 89
Basic Windows by Syscom	PC	\$ 95
BetterBASIC	PC	\$ 129
Exim Toolkit - full	PC	\$ 45
Finally - by Komputerwerks	PC	\$ 85
Mach 2 by MicroHelp	PC	\$ 55
QBase - by Crescent Software	MS	\$ 89
QuickBASIC	PC	\$ 69
Ouick Pak-by Crescent Software	PC	\$ 59
Stay-Res	PC	\$ 59
Turbo BASIC - by Borland	PC	\$ 69

TP2C - Translate Turbo Pascal to formatted K & R C (proposed ANSI 85 standard). Include files, in-line code, nested procedures. 95 + % PC \$ 219 successful conversion.

Free Literature Compare Products

Evaluate products. Compare competitors. Learn about new alternatives. One free call brings information on just about any programming need. Ask for any "Packet" or Addon Packet □ AI □ ADA, Modula □ BASIC □ "C" □ COBOL □ Editors
FORTH FORTRAN PASCAL UNIX/PC or Debuggers, Linkers.

Our Serv	vices:
Programmer's Referral List	Dealers Inquire
Compare Products	Newsletter
Help find a Publisher	Rush Order
Evaluation Literature FREE	Over 700 products
· BBS - 7 PM to 7 AM 617-740-2611	· National Accounts Cen

C Language-Compilers

AZTEC C86 - Commercial	PC	\$499
C86 PLUS - by CI	MS	\$359
Datalight C - fast compile, good	code,	
4 models, Lattice compatible,	Lib	
source. Dev'rs Kit	PC	\$ 77
Datalight Optimum - C	MS	\$ 99
with Light Tools by Blaise	PC	\$168
Lattice C - from Lattice	MS	\$269
Let's C Combo Pack	PC	\$ 99
Let's C	PC	\$ 59
Microsoft C 4.0- Codeview	MS	\$275
Rex - C/86 by Systems &		
Software - standalone ROM	MS	\$695
Turbo C by Borland	PC	\$ 69
Uniware 68000/10/20 Cross		
Compiler	MS	Call
C Libraries-Files		
O Elbrarios :s		

C Index by Trio/PLUS	MS	\$319
BTree by Soft Focus	MS	\$ 69
CBTREE - Source, no royalties	MS	\$ 99
CTree by Faircom - no royalties	MS	\$315
rtree - report generation	PC	\$239
dbOUERY - ad hoc, SQL-based	MS	\$129
dbVISTA - pointers, network.		
Object only - MSC, LAT,	C86	\$129
Source - Single user	MS	\$389
ID. tueneleten to librory	MC	\$200

dBx - translator to library C Servens Windows Graphics

C-Screens, windows, Gr	apıı	163
C-Scape - capture Dan Bricklin	PC	\$179
Curses by Aspen Scientific	PC	\$109
dBASE Graphics for C	PC	\$ 69
ESSENTIAL GRAPHICS - fast		\$185
GraphiC - new color version		\$285
Greenleaf Data Windows		\$159
w/source		\$289
Light WINDOWS/C-for Datalight	CPC	\$ 79
Windows for C - fast	PC	\$189
Windows for Data - validation	PC	\$319
Vitamin C - screen I/O		\$159
View Manager - by Blaise		\$179
7View - screen generator	MS	\$139

Atari ST & Amiga

We carry full lines of Manx & Lattice.

Call for a catalog, literature and solid value

THE PROGRAMMER'S SHOP " Your complete source for software, services and answers

5-D Pond Park Road, Hingham, MA 02043 Mass: 800-442-8070 or 617-740-2510 7/

RECENT DISCOVERY

dB2C Toolkit V 2.0 by Software Connection. 220 + dBIII functions in C source, file handler, windowing, interface to db_VISTA, c-tree, dBC III, MS, Lattice, Instant C. MS \$ 289 No Royalties

dBASE Language

Clipper compiler	PC	Call
dBASE II	MS	\$329
dBase III Plus	PC	\$429
dBASE III LANPack	PC	\$649
DBXL Interpreter by Word Tech	PC	\$139
FoxBASE + - single user	MS	\$349
Quick Silver by Word Tech	PC	\$439

dBASE Support

dBase Tools for C	PC	\$ 65
dBrief with Brief		Call
DBC ISAM by Lattice	MS	Call
dFlow - flowchart, xref	MS	Call
Documentor - dFlow superset	MS	Call
Genifer by Bytel-code generator	MS	\$299
QuickCode III Plus	MS	\$239
Tom Rettig's Library		\$ 89
UI Programmer - user interfaces	PC	\$249

Fortran & Supporting

50:More FORTRAN	PC	\$ 95
Forlib + by Alpha		\$ 59
I/O Pro - screen development		\$129
MS Fortran - 4.0, full '77		\$279
No Limit - Fortran Scientific	PC	\$109
PC-Fortran Tools - xref, pprint	PC	\$165
RM/Fortran		Call
Scientific Subroutines - Matrix	MS	\$129

Multilanguage Support

BTRIEVE ISAM	MS	\$185
BTRIEVE/N-multiuser	MS	\$455
Flash-Up Windows	PC	\$ 79
GSS Graphics Dev't Toolkit	PC	\$375
HALO Development Package	MS	\$389
HALO Graphics	PS	\$209
Informix 4GL-application builder	PC	\$789
Informix SQL - ANSI standard	PC	\$639
NET-TOOLS - NET-BIOS	PC	\$129
Opt Tech Sort - sort, merge	MS	\$ 99
PANEL	MS	\$215
Pfinish - by Phoenix	MS	\$229
Polyboost - speed I/O, keyboard	PC	\$ 69
Prime Factor FFT - 8087/287	PC	\$145
PVCS Corporate-source control	MS	\$309
PVCS Personal	MS	\$109
QMake by Quilt co.	MS	\$ 79
Report Option - for Xtrieve	MS	\$109
Screen Machine	PC	\$ 59
Screen Sculptor	PC	\$ 9:
SRMS - new version	MS	\$159
Synergy - create user interfaces	MS	\$37:
VXM - multi-env. link	MS	\$19:
Xtrieve - organize database	MS	
ZAP Communications - VT 100	PC	\$ 8

FEATURE

C Worthy Interface Library - Complete, tested human interface for MSC, Lattice or Turbo C. Full screens, Windows, DOS, Error handling, Menus, Messages Source separate, no royalties. PC \$249

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add \$3/item.

OF INTEREST

(continued from page 142)

venerable A> prompt seemed a bit much, but later the same day we found another vendor doing likewise. EGA-16 is \$24.95, and ClickStart is \$79.95. Reader Service No. 28.

hDC Computer Corp. 8405 165th Ave. NE Redmond, WA 98052 (212) 475-5550

Modems

Prices of 2,400-bps modems are coming down a bit. **Okidata** has announced two new 2,400-bps modems

at \$599 (external) and \$549 (internal), with automatic adaptive equalization to ameliorate the problems of line noise. Reader Service No. 29.

Okidata 532 Fellowship Rd. Mount Laurel, NJ 08054 (609) 235-2600

The **Zoom**/Modem PC 2400 HC is an internal 300/1,200/2,400-bps IBM PC, IBM PC/XT, IBM PC/AT, and compatible modem with a suggested retail price of \$199. It supports Bell 103a,

212a, and CCITT v.22 bis protocols and uses the Hayes AT command set. Reader Service No. 30.

Zoom Telephonics Inc. 207 South St.

Boston, MA 02111 (617) 423-1072

Displays

Jeff Duntemann, editor of Borland's new *Turbo Techniques*, often nags software vendors about support for full-screen displays. Jeff owns a Genius 15-inch full-page display and has the radical idea that software developers should not penalize him for owning a good monitor. **Micro Display Systems** announced at Comdex a line of 19-inch Genius full-screen displays using the TI 34010 graphics coprocessor. Reader Service No. 31.

Micro Display Systems Inc. 1310 Vermilion St.

P.O. Box 455 Hastings, MN 55033 (612) 437-2233

Cornerstone Technology has introduced the Vista 1600, a \$2,195 19-inch monitor for the Macintosh II that displays 1,600 × 1,280-resolution, the Mac's max. Its noninterlaced screen has a 200-MHz bandwidth and refreshes at 67 Hz. The monitor comes with a NuBUS controller card that occupies one slot in the Mac II.

Cornerstone has also announced a version of the monitor for PCs and 386 machines for \$2,395. Reader Service No. 32.

Cornerstone Technology 175A East Tasman Dr. San Jose, CA 95134-1620 (408) 433-1600

Hitachi has announced more hi-res monitors, including a 20-inch $1,280 \times 1,024$ -model with RGB input and various scanning frequencies for \$3,995 suggested retail. Reader Service No. 33.

Hitachi Sales Corp. of America 401 West Artesia Blvd. Compton, CA 90220 (213) 537-8383

Amdek has announced a family of color and monochrome displays compatible with the PS/2's VGA spec. Models 432 (monochrome) and 732

Never Miss A Compile Again!

BSW-Make, our retargetable *make* utility, speeds software development by automating the chore of rebuilding complex software products after an editing session. No more missed compiles! No more wholesale "just in case" recompilations of the whole product! BSW-Make insures that the minimum set of compilations, assemblies, and links required to correctly update your software are performed after each edit. A major timesaver!

- Syntax compatible with UNIX make
- Works with any compiler, assembler, or linker
- Macro facility for parameterized builds
- Indirect command file generation facility overcomes operating system command length limitations
- MS-DOS/PC-DOS version only \$89.95
- VAX/VMS version from \$299.95
- Not copy protected
- Unconditional 30-day guarantee try it at no risk!

for free product information, call (617) 367-6846

Ask for Department D2

The Boston Software Works, Inc.

120 Fulton Street, Boston, MA 02109

CIRCLE 384 ON READER SERVICE CARD

SOFTWARE ENGINÉERING COMES OF AGE.

ANNOUNCING LOGITECH MODULA-2 VERSION 3.0

LOGITECH Modula-2 is the most powerful implementation available for the PC. tools have come together in one program or a complex project, Version 3.0 you can write more reliable, maintainable, better docu-

> FREE TURBO PASCAL TO LOGITECH MODULA-2 TRANSLATOR

NEW IMPROVED

be lost at debug time without the right debugging tools. With the powerful Logitech Modula-2 Debuggers you can debug your code fast, and dramatically improve your overall



project throughput. The Post Mortem while the dynamic,

Run Time Debugger monitors the execution of a program with user-defined break points. With their new, mouse these powerful debugging tools are a pleasure to use.

NEW, INTELLIGENT LINKER

Links only those routines from a particular module that you need, so you eliminate unreferenced routines and produce smaller, more compact



V. 3.0 Compiler Pack
Compiler in overlay and fully linked form.
Linkable Library, Post Mortem Debugger, Point Editor

☐ LOGITECH Modula-2 3.0 Toolkit Library sources, Linker, Run Time Debugger, MAKE, Decoder, Version, XRef. Formatter

■ LOGITECH Modula-2 V. 3.0 Development System Compiler Pack plus Toolkit

Turbo Pascal to
Modula-2 Translator

FREE With Compiler Pack or Development System

Window Package
Build true windowing into your
Modula-2 code.

Upgrade Package Call LOGITECH for information or to receive an order form.

Add \$6.50 for shipping and handling. California residents add applicable sales tax. Prices valid in U.S. only.

Total Enclosed \$______

Total Enclosed \$ □ VISA □ MasterCard □ Check Enclosed

Card Number	Expiration Date
Signature	
Name	
Address	
City	State
Zip	Phone

NEW, IMPROVED COMPILER Faster and more flexible. Now

its DOS linker compatible object files (.OBJ) can be linked with FORTRAN and ASSEMBLER development and put the power of LOGITECH Modula-2 to work for you right now. Fully

> LONGINT and LONGSET, which provides large set support including SET.

efficient code generation.

NEW EDITOR

Our new, mouse based editor is fully integrated, easy to learn, fast and easy color support make it easy to manage screen at one time. You'll love using it — with or without a mouse.

Dealer & Distributor pricing.

800-231-7717 In California: 800-552-8885



6505 Kaiser Drive, Fremont, CA 94555 Tel: 415-795-8500

In Europe: LOGITECH, Switzerland Tel: 41-21-87-9656 Telex 458 217 Tech Ch

In the United Kingdom: LOGITECH, U.K. Tel. 44908-368071 Fax: 44908-71751

Turbo Pascal is a registered trademark of Borland International. VAX and VMS are registered trademarks of Digital Equip



USING THE WRONG LANGUAGE CAN BE MURDER. SPEAK SMALLTALK/V.



Let's talk languages. Programming languages like Turbo Pascal, C or Basic can be killers. To many, they're foreign, complex, and generally intimidating. Mistakes can be deadly.

With Smalltalk/V, you have an elegantly simple solution that puts the power and majesty of

a major AI programming language on your PC or compatible. It makes no difference if you're an experienced programmer or just getting started. Smalltalk/V gives you an easy-to-use and flexible programming tool.

This is the same language used by leading software companies for their new product development. There are sound reasons for this. Smalltalk/V offers a totally integrated programming environment using the premier object-oriented language. You use natural language rather than complex

programming codes. It puts Macintosh-type graphic features on a PC including overlapping windows, bit-mapping, pop-up menus, and a mouse interface. More than mere window dressing, Smalltalk/V delivers fully interactive windows that are easy to build and quick to modify.

But don't just take our word on it. Hear what the experts have to say:

"This is the real thing folks. A super Smalltalk like this turns your PC into a hot workstation. It's fantastic... Highly recommended."

John Dvorak Contributing Editor PC Magazine

"The tutorial provides the best introduction to Smalltalk available."

Dr Andrew Be

Dr. Andrew Bernat
AI Expert Magazine

CIRCLE 127 ON READER SERVICE CARD

"Smalltalk/V is the highest performance object-oriented programming system available for PCs."

Dr. Piero Scaruffi Chief Scientist Olivetti Artificial Intelligence Center

Today, thousands of professionals, scientists and engineers are using Smalltalk/V to solve both simple and expert problems. Giving them a new dimension in computer applications for their PC.

Put new life into your PC by calling toll free 1-800-922-8255 and ordering Smalltalk/V today. Smalltalk/V by Digitalk, Inc., 9841 Airport Blvd., Los Angeles, CA 90045. (213) 645-1082.

Smalltalk/V comes with 10 starter applications including Prolog and each Application Pack adds several more. All source code is included. Supports 640 x 480 color graphics with color extension pack.

Smalltalk/V requires DOS and 512K RAM on IBM PC/AT/PS or compatibles and a CGA, EGA, Toshiba T3100, Hercules, or AT&T 6300 graphic controller. A Microsoft or compatible mouse is recommended. Not copy protected.

Turbo Pascal is a trademark of Borland Internation. IBM, IBM PC/AT/PS are trademarks of International Business Machines Corporation. Macintosh is a trademark of Apple Computer, Inc

TO ORDER CALL 1-800-922-8255 TODAY.

60-DAY MONEY-BACK GUARANTEE*
Send check, money order, or credit card information to: Digitalk, Inc., 9841 Airport Boulevard, Los Angeles, CA 90045.
Credit Card VISA Mastercard
Card number:

Expiration date:

Name:

Street Address:

City/State/Zip:

Smalltalk/V

digitalk inc.

Smalltalk/V	\$99.95
RS-232 Communications Application Pack	\$49.95
EGA/VGA Color Extension	on
Pack	\$49.95
"Goodies" Application Pack	\$49.95
SPECIAL OFFER: Smalltalk/V and all	
3 packs only	\$199.95
Shipping and handling (Outside North	\$5.00
America	\$15.00)
California residents add applicable sales tax	

TOTAL *Unconditional 60-day money-back guarantee. Simply return to Digitalk, Inc. and your refund will be immediately forwarded to you.

SAVE YOUR PC FROM EARLY RETIREMENT.



GET HAUPPAUGE'S NEW 386 MOTHERBOARD.

386 SPEED-ONLY \$1,495

Give your PC a new lease on life! With our industry first 386 MotherBoard, your PC, PC/XT or compatible will revel in speeds equal to the Compaq DeskPRO 386. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that will knock you off your rocker.

To keep retirement at bay, our 386 MotherBoard is compatible with the PC/AT (BIOS and I/O) — allowing you to run the new generation of DOS, OS/2. We've also included a 16-bit expansion slot that accommodates the latest I/O expansion card. No accelerator card can give you so much versatility.



With 386 power and true AT software compatibility, your business, desktop publishing and engineering applications will get a boost to boast about! **Technical Features** = 16 MHz 80386 = 1 Megabyte of 100 nsec 4-way interleaved RAM = PC/AT compatible I/O and BIOS for support of OS/2 = Seven 8-bit expansion slots = Two 16-bit expansion slots = One 32-bit RAM expansion slot = Optional 16 MHz 80387 math coprocessor (\$695)

Put the power of the 386 into your IBM PC for 1/4 the cost of a 386 computer. And put off your PC's retirement. For more information on our easy-to-install Motherboard, call 1 (800) 443-6284. In New York, call (516) 360-3827. Hauppauge Computer Works, Inc.

358 Veterans Memorial Highway,
Commack, New York 11725
Trademarks: IBM.PCAT.

Hauppauge!

OF INTEREST

(continued from page 144)

(color) display 640×480 (VGA), $640 \times$ 400 (CGA), or 640 × 350 (EGA) pixels and sell for a suggested retail price of \$245 and \$625, respectively. Reader Service No. 34. Amdek Corp. 1901 Zanker Rd. San Jose, CA 95112 (408) 436-8570

Tseng Labs has announced a new VLSI chip for IBM VGA graphics, for which the company expects to show a prototype this month. The ET3000 chip will support all VGA graphics as well as MGA, CGA, and EGA. Reader Service No. 35.

Tseng Laboratories Inc. 10 Pheasant Run Newtown, PA 18940 (215) 968-0502

CAD/CAM/CAE

Vector Automation claims that its CADMAX Version 3.0 software for 386 workstations, formerly available on MicroVAX II, is the first CAD software to take full advantage of the 386. The

price is \$3,350 for a 2-D version. Reader Service No. 36.

Vector Automation Inc. Village of Cross Keys Baltimore, MD 21210 (301) 433-4200

Modgraph's Pegasus is a graphics card that supports EGA (or CGA, MGA, or Hercules) and high-resolution graphics on one monitor of an IBM PC, IBM PC/XT, IBM PC/AT, or compatible computer, for CADD work and the like. The hi-res support includes an 82786 graphics engine, there is a chip set supporting EGA, and the board discerns which mode you want. Reader Service No. 37.

Modgraph 149 Middlesex Turnpike Burlington, MA 01803 (617) 229-4800

The Great Western Software Company has a companion software product to AutoCAD called Auto-Board System II for use in the production of printed circuit boards. This is TGWSC's third automatic routing package for microcomputers; it's been at it for a while. Reader Service No. 38.

The Great Western Software Co. 207 West Hickory St., Ste. 202 Denton, TX 76201 (817) 383-4434

The MSA Group, developer of a 2-D CAD system for PCs, took the name of its product, TurboCAD, seriously, and, following the Borland lead, cut the product's price from \$395 to \$99 for all modules. Reader Service No. 39. MSA Group 12021 Wilshire Blvd., Ste. 370 West Los Angeles, CA 90025

Computervision has announced Version 3.0 of its PC-based 3-D CAD/ CAM package, Personal Designer. New features include multiple views, improved dimensioning capability, and an undo feature. Reader Service No. 40.

Computervision Corp. 100 Crosby Dr. Bedford, MA 01730 (617) 275-1800

(213) 473-8711

DDJ

Australia's Finest C Compiler



plus shipping

HI TECH C Compiler

- Complete production quality compiler
- Smallest, fastest code from any compiler
- High performance C Compiler for the Z80, 68000, 65816, and 8086 processors
- Runs on CP/M-80, PC-DOS, MS-DOS, CP/M-86, CONCURRENT CP/M, ATARI ST and APPLE II as
- Now in use at thousands of sites worldwide, including Australian Government and large institutions.
- Excellent user interface
- ROM code is supported and it includes a macro assembler, linker, librarian, object code converter, cross reference utility and full library source code. The 8086 compiler supports large and small memory models and

plus shipping

Cross Compilers

 Run under MS-DOS, UNIX, and CP/M-86 and produce code for the 68000, 8086/286, 65816, 8096 and Z80 processors. Each compiler includes an assembler, linker, librarian, object code converter and cross reference utility.



The Cutting Edge

Order from: SOFTFOCUS 1343 Stanbury Drive, Oakville ONTARIO Canada L6L2J5 (416) 825 0903 or (416) 844 2610

> U.K. Greymatter (0364) 53 499 Australia HITECH SOFTWARE P.O. Box 103 Alderley 4051 (07) 38 6971

CIRCLE 376 ON READER SERVICE CARD

HARD FACTS:

We're Programmer's Connection, your best one-stop source for quality programmer's development tools for IBM personal computers and compatibles. Here are some important facts you should know about us and other dealers in our industry.

FACT:

FREE Shipping. Shipping to U.S. customers is FREE via UPS Ground. If you want your order shipped via an express service, we'll only charge you the shipping carrier's standard rate with no special fees. Some dealers charge extra for shipping and then add rush charges for shipping via express services. Others may advertise "free" shipping but make up for it by charging extra handling fees.

FACT:

Credit Cards. We'll charge your credit card only when we actually ship your order. Some dealers would charge your credit card at the time you place your order. This could leave you waiting for your shipment for weeks or months while they use your money interest-free.

FACT:

Discounts. We discount all software products — even special order items. Every product in our advertised price list is shown with its list price and discounted price. We want you to know exactly how much you'll save on every product. We don't try to fool you by discounting some products and charging full retail for others.

FACT:

Consistent Prices. We extend the same current prices to every customer regardless of where they see our ad. Some dealers vary prices in different ads and then ask you to mention which one you saw. This technique allows them to charge you the highest prices possible.

FACT:

No Hidden Charges. The discount prices you see on the next two pages are all you pay. We don't charge extra for UPS Ground shipping, credit cards, COD orders, purchase orders, sales tax (except Ohio) or special handling (except for non-Canadian international orders).

FACT

Guarantees. We offer FREE 30-day no-risk return guarantees and 30-day evaluation periods on most of our products. Some dealers have no return options while others often charge restocking fees of 15% or more.

FACT

Quality Products. Our product line consists of hundreds of high quality software development tools specifically for IBM Personal Computers and compatibles. While some dealers try to carry every software product ever written, we carry only those that meet our very high standards for quality and value.

FACT:

Latest Versions. The products we carry are the latest versions and come with the same manufacturer's technical support as if buying direct. While some dealers may participate in the software gray market, we're authorized to sell every product we carry.

FACT:

Large Inventory. We have one of the largest inventories of programmer's development products in the industry. Most orders are shipped within 24 hours. And if we don't have a product in stock, we'll get it for you fast.

Turn the page for our product list and ordering information.

FACT

Meticulous Packaging. We'll give your shipment the extra protection needed to reach you in the best possible condition. First we'll protect your products from moisture by wrapping them in plastic. Then we'll insulate your box with high quality bubble-wrapping instead of the messy styrofoam chips that many other dealers use. Finally, we'll double-tape your box for extra strength.

FACT:

Independence. Since we're not directly affiliated with any software publisher or developer, we can give you sound, unbiased advice. Unlike some dealers who have a special interest in promoting only certain products, we'll give you an objective look at the products we carry.

FACT

Noncommissioned Staff. Our courteous sales staff is always ready to help you. And if you aren't sure about your needs, our knowledgeable technical staff can give you sound, objective advice. Because they are noncommissioned, you won't be pressured into making a purchase.

As you can see, we're different from the other dealers in our industry. Our customers keep coming back because we consistently provide the highest quality service and the lowest prices. So call us today and experience the differences for yourself.



CIRCLE 129 ON READER SERVICE CARD

			Sidekick & Traveling Sidekick	125 85	85 57	dos utilities	00	
ai - expert systems	List	Ours	Traveling Sidekick	70 100	45 64	Command Plus by ESP Software	80 75	69 62
1st-CLASS by Programs in Motion	495	399	Superkey	100	64	MicroHelp Utilities by MicroHelp	59 75	49 CALL
EXSYS Development Software by EXSYS	395 600	309 469	Turbo BASIC Database Toolbox New Turbo BASIC Editor Toolbox New	100	64 64	OPAL Shell Language by Software Factory	99	89 59
Logic-Line Series All varieties by Thunderstone	CALL	CALL	Turbo BASIC Telecom Toolbox New Turbo C Compiler (Call for support products)	100	64 64	Taskview by Sunny Hill Software	80	55
ai - lisp language	ADE	CALL	Turbo Database Toolbox	70 70	41 41	essential products		
GCLISP Golden Common LISP by Gold Hill	495 1190	CALL	Turbo Gameworks Toolbox	70 70	41 41	C Utility Library	185	119 189
Microsoft LISP Common LISP	250 CALL	149 CALL	Turbo Graphix Toolbox	300	219	Essential Comm Library Software Only	185	125
TransLISP PLUS from Solution Systems	195	125	Turbo Lightning	100	64 64	Breakout Debugger Only Any language	125 250	89 183
ai - prolog language	1095	979	Turbo PASCAL and Tutor	125 100	85 64	forth language		
Arity Combination Package Expert System Development Pkg	295	229	Turbo Tutor	40 100	24 64	CFORTH Native Code Compiler by LMI	300	229
File Interchange Toolkit	50 650	569	Turbo PROLOG Compiler	100	64	Forth/83 Metacompiler Specify Target	750 150	599 109
Screen Design Toolkit	50 295	229	Word Wizard Word Wizard and Turbo Lightning	70 150	47 94	PC/Forth+ by Laboratory Microsystems	250 100	199 74
Arity PROLOG Interpreter	295	229	ccompilers			Enhanced Graphics Support	200	148 74
Arity Standard Prolog	95 CALL	CALL	C86PLUS by Computer Innovations	497 209	375 184	Interactive Symbolic Debugger	100	74 74
MPROLOG Language Primer LOGICWAREMPROLOG P500 by LOGICWARE	50 495	45 395	DeSmet C w/Debugger & Large case DeSmet C w/Debugger only	159	138	Native Code Optimizer	100	74
MPROLOG P550 by LOGICWARE Turbo PROLOG by Borland Intl	220 100	175 64	Eco-C Complete System by Ecosoft	140 500	119 265	UR/Forth by LM/	350 500	279 395
Turbo PROLOG Toolbox by Borland Intl	100	64	Mark Williams Let's C w/csd	125 450	99 269	fortran language		
ai - smalltalk language			Microsoft QuickC Compiler New Optimum-C by Datalight	99 139	63 105	50 MORE: FORTRAN by Peerless Scientific	125	95
Smalltalk/V EGA/VGA Color Option	99 50	84 45	Turbo C Compiler by Borland Uniware 68000/10/20 Cross Compiler by SDS	100	64	ACS Time Series by Alpha Computer Service AUTOMATED PROGRAMMER by KGK Automated New	495 CALL	CALL
Goodies Diskette	50 50	45 45	19 BB (19 BB) 10 BB (19 BB) 전 10 BB (19 BB) 15 B	995	829	Essential Graphics by Essential Software For-Winds by Alpha Computer Service	250 90	183 69
ai - texas instruments	00		Cinterpreters C-terp by Gimpel, Specify compiler New Version	300	219	Forlib-Plus by Alpha Computer Service	70 125	109
Arborist Decision Tree Software New	595	519	C Trainer with Book by Catalytix	122 500	87 369	FORTLIB by Sutrasoft	95	85
PC Scheme Lisp	95 495	84 435	Instant C by Rational Systems New Version Introducing C by Computer Innovations	125	99	GRAFLIB by Sutrasoft	165 175	138 159
Personal Consultant Image	495 995	435 869	Run/C by Age of Reason	120 250	69 145	HALO Graphics by Media Cybernetics I/O PRO w/No Limit Library by MEF	300 250	205
Personal Consultant Plus	2950	2589	cutilities			Microcompatibles Combo Package	240 135	215 117
Personal Consultant Runtime	95	84	C++ by Guidelines w/version 1.1 kernel	195	172	Grafmatic	135	117
ada language AdaVantage by Meridian Software Systems New	795	735	Csharp Realtime Toolkit by Systems Guild New c-tree & r-tree Combo by FairCom	600 650	539 519	Microsoft FORTRAN w/CodeView	450 129	269 109
AdaVantage Utility Packages New DOS Environment Package New	CALL 50	CALL	c-tree ISAM File Manager r-tree Report Generator	395 295	315 239	Numerical Analyst by MAGUS	295 295	249 215
Janus/ADA C Pak by R&R Software	95	84	Data Windows by Magus Inc New	245 595	209 499	PLOTHP by Sutrasoft	175	159
Janus/ADA D Pak by R&R Software Janus/ADA ED Pak by R&R Software	1250 395		with Source Code	350	299	RM/FORTRAN by Ryan-McFarland	595 450	399
apl language			With Source Code New Flash-up Windows by Software Bottling	550 90	469 78	Scientific Subroutine Lib by Peerless	175 295	135 235
APL*PLUS/PC by STSCAPL*PLUS/PC Spreadsheet Mgr by STSC	595 195		GraphiC Color version by Sci Endeavors	350 175	282 159	Statlib.GL: by Peerless New Version Statlib.TSF: by Peerless New Version	295 295	239 239
APL*PLUS/PC Tools Vol 1 by STSC	295	199	HALO Graphics by Media Cybernetics	300 595	205 389	Strings & Things by Alpha Computer Service	70	45
APL*PLUS/PC Tools Vol 2 by STSC	85 450	329	HALO Development Pkg for Microsoft	195	129	greenleaf products		
Financial/Statistical Library by STSC	275 95		PANEL Forms Management by Roundhill	295 129	215 95	Greenleaf Comm Library	185 225	125 155
STATGRAPHICS by STSC	795		PANEL Plus by Roundhill	495 139	395 99	Greenleaf Data Windows Library	395	249
assembly language			PLOTHP by Sutrasoft	175	159	Greenleaf Functions	185	125
386 ASM/LINK Cross Asm by Phar Lap	495 100		RTC PLUS Fortran to C by Cobalt Blue Scientific Subroutine Library by Peerless	450 175	399 135	help utilities	125	99
ASMLIB Function Library by BC Assoc	149 395		TE Text Editor source by Sub Systems New Vitamin C by Creative Programming	95 225	85 158	HELP/Control by MDS	125 149	99
Cross Assemblers Various by 2500 AD	CALL	CALL	VC Screen Forms Designer Zview by Data Mgmt Consultants	100 245	79 139	SoftScreen/HELP by Dialectic Systems	195	149
EZASM by C Source New Microsoft Macro Assembler New Version	150	93	cobol language	240	100	lattice products	500	205
Norton Utilities by Peter Norton	100 150	99	COBOLspli by Flexus	395	329	Lattice C Compiler ver 3.2 from Lattice	900	265 495
Turbo Debugger by Speedware New Turbo Editasm by Speedware	89 99		FPLIB for Realia COBOL by BC Associates	149	129	C Cross Reference Generatorwith Source Code	50 200	37 139
Visible Computer: 80286 New Visible Computer: 8088 by Software Masters		89	Microsoft COBOL See Microsoft Section PCDT by Pro-Code New	995	895	C-Food Smorgasbord Function Library	150 300	95 179
basic language	00	04	Realia COBOL with RealMENU New Version	1145	899 783	C-Sprite Source Level Debugger	175	119
87 Software Pak by Hauppauge	180		Realia COBOL New Version RealCICS	995	783	Curses Screen Manager	125 250	85 169
db/Lib for QuickBASIC by AJS Publishing New EXIM Services Toolkit by EXIM	99		RM/COBOL by Ryan-McFarlandRM/COBOL 85 by Ryan-McFarland	950 1250	CALL	dBC Specify dBC II or dBC III	250 500	169 356
Finally by Komputerwerks	99		SCREENIO by Norcom New screenplay for COBOL by Flexus	400 175	379 129	dBC III Plus	750 1500	594 1184
MACH 2 by Micro Help Microsoft QuickBASIC\$20 Rebate Offer	99	63	css products			LMK Make Facility RPG II Combo All three items below	195 1100	138
QBase Relational Database by Crescent New Quick-Tools by BC Associates	130	109	Combo Package by Custom Software Systems New	199	175	RPG II Compiler No Royalties	750	625
QuickPak by Crescent Software	125	99	PC/SPELL Spelling Checker	49 49	45 45	RPG II SEU Screen Entry Utility	250 250	199
Screen Sculptor by Software Bottling	125		PC/VI vi Editor	149	99	RPG II Screen Design Aid Utility	350 120	
True Basic w/Run-time Special Price	200	99	debuggers & profilers	195	129	SideTalk Resident Communications	120	88
True Basic	100	79	386 DEBUG Cross Debugger by Phar Lap	175	115	SSP/PC Scientific Subroutine Library Text Management Utilities	350 120	88
Various Utilities	100		Codesmith-86 by Visual Age	125	98 79	TopView Toolbasket Function Library with Source Code	250 500	178 356
blaise products			MiniProbe by Atron Periscope I with Board by Periscope	395 345	369 289	metagraphics products	5	
ASYNCH MANAGER Specify C or Pascal	175		Periscope II with NMI Breakout Switch	175 145	139 105	LightWINDOW/C for Datalight C	95 95	79 79
LIGHT TOOLS for Datalight C Special Price PASCAL TOOLS	100	65	Periscope III 8 MHz version Periscope III 10 MHz version	995 1095	799 899	FontWINDOW	275	229
PASCAL TOOLS 2	100	79	The PROFILER with Source Code by DWB	125	89	MetaWINDOW No Royalties	195 275	229
PASCAL TOOLS & TOOLS 2	50	45	TURBOsmith Source debugger for Turbo Pascal	69 60	59 51	TurboWINDOW/C for Turbo C	95 95	79 79
TURBO ASYNCH PLUS	100	CALL	disk utilities					
TURBO POWER TOOLS PLUS	100	79	Back-It by Gazelle Systems Disk Optimizer by Softlogic Systems	100	89 55	micro focus products Micro Focus Level II COBOL w/Animetor	495	
borland products		.00	FASTBACK by 5th Generation Systems Vcache by Golden Bow Systems New	179	129 47	Level II COBOL Level II Animator	349 195	279
EUREKA <i>Equation Solver</i> Reflex & Reflex Workshop	16		Vopt by Golden Bow Systems New Vteature by Golden Bow Systems New	50	47	Micro Focus Level II COBOL/ET for UNIX	CALL 149	CALL
Reflex Data Base System	150	89	Vfeature Deluxe by Golden Bow Systems New	120	111	Micro Focus Professional COBOL	2000	1595
Reflex Workshop	71	0 45	XenoCopy-PC by XenoSoft	80	69	Micro Focus VS COBOL/XENIX	1495	1195

Micro Focus Support Products:	405	205	PolyLibrarian Library Manager	99	89
COBOL/IQ Ad hoc Report Writer	495 995	395 795	PolyLibrarian II Library Manager	149 149	129 129
FORMS-2 SOURCEWRITER	295	235	PolyShell	149	129
SOUNCEWNITCH	995	795	Polytron C Beautifier	50 219	45 185
microport products			PolyXREF Une language only	129	109
System V/386 Combination New 386 Runtime System New	799 199	699 169	PVCS Corporate Version Control System	395 149	329 129
386 Software Development System New	499	429	program mgmt utilities		
Text Preparation System New 386 Unlimited License Kit New	199 249	169 209	Interactive EASYFLOW by Haventree	150	125
DOSMerge386 Run DOS and UNIX together New	CALL	CALL	PrintQ by Software Directions	89	84
System V/AT Combination	549 199	465 169	Quilt Computing Combo Package	250 99	199 79
AT Software Development System	249	209	SRMS Software Revision Mgmt System	185	159
Text Preparation System	199 249	169 209	Source Print by Aldebaran Labs TLIB Version Control System by Burton	97 100	75 89
DOSMerge286 Run DOS and UNIX together New	149	129	Tree Diagrammer by Aldebaran Labs	77	67
microsoft products			raima products		
Microsoft BASIC Compiler for XENIX	695	419	dbQUERY Single-User Query Utility	195	129
Microsoft BASIC Interpreter for XENIX	350 450	209 269	Single-User with Source Code	495 495	389 389
Microsoft COBOL Compiler with COBOL Tools	700	429	Multi-User with Source Code	990	699
for XENIX	995 450	609 269	dbVISTA Single-User DBMS	195 495	129 389
Microsoft FORTRAN Optimizing Compiler Codeview	695	419	Multi-User	495	389
Microsoft LISP Common USP	50 250	36 149	Multi-User with Source Code	990	699
Microsoft MACH 10 with Mouse & Windows	549	369	sco products		
Microsoft MACH 10 Board only	399 150	279 93	Complete XENIX System V by SCO	1295 595	994 499
Microsoft Macro Assembler New Version Microsoft Mouse for IBM PS/2 New	175	114	Operating System Specify XT or AT	595	499
Microsoft Mouse Bus Version	175	114	Text Processing Package	195 595	144 449
Microsoft Mouse Serial Version	195 300	124 179	SCO Professional 1-2-3 Workalike for XENIX	795	595
Microsoft Pascal Compiler	300	179	SCO XENIX-NET	595	495
for XENIX	695	419 63	softcraft products		
Microsoft QuickC	99	63	Btrieve ISAM Mgr with No Royalties	245	184
Microsoft Sort	195	125 63	Xtrieve Query Utility	245 145	184 99
Microsoft Windows Development Kit	500	299	Btrieve/N for Networks	595	454
Microsoft Word	450	269	Xtrieve/N	595 345	454 269
modula-2 language			text editors		
EXE2LNK MASM Interface by PMI New Macro2 Macro preprocessor by PMI New	49 89	45 79	Brief & dBrief Combo from Solution Systems	275	CALL
ModBase by PMI New MODULA-2 Apprentice Pkg by LOGITECH	89	79	Brief	195	CALL
MODULA-2 Apprentice Pkg by LOGITECH	99 99	79 79	dBrief Customizes Brief for dBASE III	95 195	CALL 147
MODULA-2 Magic Pkg by LOGITECH	299	239	KEDIT by Mansfield Software	125	98
MODULA-2 Window Pkg by LOGITECH	199	39 159	Micro/SPF by Phaser Systems	175 450	139 269
Repertoire by PMI	89	75	PC/VI by Custom Software Systems	149	105
mouse products			SPF/PC by Command Technology Corp	CALL 185	CALL 128
LOGIMOUSE BUS with PLUS Pkg by LOGITECH	119	98	turbo pascal utilities		
with PLUS & PC Paintbrush	149	119 153	ALICE Interpreter by Software Channels	95	66
with PLUS & CAD Software	189 219	179	DOS/BIOS & Mouse Tools by Quinn-Curtis	75	67
LOGIMOUSE C7 with PLUS Pkg, Specify Connector	119	98 119	Flash-up Windows by Software Bottling	90 69	78 55
with PLUS & PC Paintbrush	149 189	153	MetraByte D/A Tools by Quinn-Curtis	100	89
with PLUS & CAD & Paint	219	179	Science & Engrg Tools by Quinn-Curtis	75 125	67 91
microsoft mouse see microsoft Section			Speed Screen by Software Bottling	35	32
other languages			System Builder by Royal American IMPEX Query Utility	150 100	129 89
ACTOR by Whitewater Group New CCS MUMPS Single-User by MGlobal	495 60	419 50	Report Builder TDebugPLUS by TurboPower Software	130	115
CCS MUMPS Single-User Multi-Tasking	150	129	TDebugPLUS by TurboPower Software New	60 80	49 69
CCS MUMPS Multi-User	450 189	359 165	Turbo EXTENDER by TurboPower Software	85	64
Pascal-2 by Oregon Software	395	325	Turbo OPTIMIZER by TurboPower Turbo OPTIMIZER with Source Code	75 125	65 108
Personal REXX by Mansfield Software SNOBOL4+ by Catspaw	125 95	99 80	Turbo Professional by Sunny Hill	70	45
	33	00	Turbo.ASM by BC Associates New TurboHALO from IMSI	100 129	89 98
other products	100		TurboPower Utilities by TurboPower	95	78
Carbon Copy by Meridian Technology New Dan Bricklin's Demo Pgm by Software Garden	195 75	179 57	TurboRef by Gracon Services	50 69	45 59
Dan Bricklin's Demo Tutorial	50	45	Universal Graphics Library by Quinn-Curtis	150	119
Fast Forward by Mark Williams New First Publisher with Mouse from Logitech New	70 CALL	59 CALL	wendin products		
Informix All Varieties by Informix	CALL	CALL	Operating System Toolbox	99	79
Instant Replay by Nostradamus	150 99	CALL 89	PCNX Operating system	99	79
MKS Toolkit w/vi Editor by MKS New Version	139	114	Wendin-DOS Multitasking DOS New	99	79 85
MicroTEX Typesetting from Addison Wesley	295 149	CALL 129	XTC Text Editor w/Pascal source	99	75
OPT-Tech Sort by Opt-Tech Data Proc	149	99	xenix/unix products		
PC/TOOLS by Custom Software	49 79	45 59	Btrieve ISAM File Mgr by SoftCraft	595	454
			C-terp by Gimpel, Specify compiler	498 395	379 329
Phoenix products	195	108	dBx with Library Source by Desktop Al	550	469
Pasm86 Macro Assembler version 2.0	145	99	Desqview from Quarterdeck	100 399	85 349
Pfantasy Pac Phoenix Combo	995 395	595 209	DOSIX User Version by Data Basics	199	179
Pfinish Execution Profiler Pfix86plus Symbolic Debugger	395	209	Informix All Varieties by Informix	CALL	CALL
PforCe Specify C Compiler	395 395	209 209	Microport Products See Microport Section Microsoft Products See Microsoft Section		
Plink86plus Overlay Linker	495	275	PANEL Plus by Roundhill Computer Systems	795	535
Pmaker Make Utility	125 195	78 108	REAL-TOOLS Binary Version by PCT	149 399	89 289
Pre-C Lint Utility	295	154	Complete Source Version	999	729
Ptel Binary File Transfer Program	195	108	RM/COBOL by Ryan-McFarland	1250 750	CALL
polytron products			SCO Products See SCO Section		
PolyBoost The Software Accelerator	80	64			
PolyDesk III	99 50	72 42	Call or write for our FREE comprehensive pri	ce gui	de.
PolyDesk III CryptographerPolyDesk III Talk	50 70	42 52	® Copyright 1987 Programmer's Connection In	corpor	ated.

LOWEST PRICES

Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

FREE SHIPPING

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

CREDIT CARDS

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and authorized signature.

CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of \$100) are available to qualified US accounts only.

SALES TAX

Orders outside of Ohio are not charged state sales tax. Ohio customers please add 6% Ohio tax or provide proof of tax-exemption.

INTERNATIONAL ORDERS

Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional \$10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering

Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS

Volume orders may qualify for additional discounts. Call us for special pricing.

SOUND ADVICE

Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

30-DAY GUARANTEE

Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

MAIL ORDERS

Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

Programmer's Connection Order Processing Department 136 Sunnyside Street Hartville, OH 44632

USA	800-336-1166
CANADA	800-225-1166
OHIO & ALASKA (Collect)	216-877-3781
TELEX	9102406879
EASYLINK	62806530
INTERNATIONAL	216-877-3781

CUSTOMER SERVICE 216-877-1110 Hours: Weekdays 8:30 AM to 8:00 PM EST.



SWAINE'S FLAMES

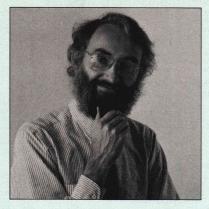
t seems that look and feel may be less of a problem for Lotus Development Corp. than compiled spreadsheets are. Several products now turn 1-2-3 spreadsheet data into executable programs, providing a legal way to share and manipulate spreadsheet data while only buying one copy of the software. What you can't do with these programs is design, or alter the design of, spreadsheets. In a company where spreadsheets are designed by one person and used by many, these products could save a lot of bucks-bucks that Lotus may feel ought to be flowing its way. Should be interesting.

Personally, I find it weird to think of 1-2-3 as a programming environment.

Speaking of weird programming environments, terminate-and-stay-resident implementations of BASIC and other programming languages are popping up all over the place. If you ask me, anyone who develops software in TSR space is playing with fire, but that won't stop the crazed code junkies from snorting this stuff up.

Well, you might program in the ether, but you don't mess around with the herald of Galactus. Marvel Comics has leaned on Acius about the name Silver Surfer, and the company's Mac database is now called 4th Dimension. In an unrelated development in the same paragraph, Living Videotext recently moved to Easy Street (really), and there were rumors that the company was about to be bought by Symantec. More interesting were the hints that the vendor of a very successful windowing system was being pressured by a wellknown venture capitalist to sell out to a major operating system vendor for the good of the industry.

And you thought the Ollie North Show was the best soap opera in town.



Here, for all you model programmers, is what you need to do to become a gatefold in a major computer magazine: As a result of your strategic partnership with the largest computer company, you deliver a multitasking operating system (retaining the right to license it to your partner's competitors), and when your partner lays out its plans to add competitive value to the product with connectivity extensions, you strike another strategic partnership with a leading LAN vendor, intended to give you a piece of everybody else's answer to those extensions. Meanwhile you hedge your bets on the operating system itself by striking a strategic partnership with a leading multiuser operating system vendor, and when you've got that all sewn up, you convince programmers to pay you thousands of dollars to learn how to develop the software to run under your operating system.

Cousin Corbett's Secrets of Software Success, Part II: Product Names.

The following advice from my cousin Corbett's forthcoming book is reprinted here without comment.

"Although there is no formula for successful product naming, there are a few simple rules that will help you avoid the most common mistakes.

"Try for unambiguous pronounceability. Unless you're selling sexually explicit software, people ought to be able to go into a store and ask for your product without embarrassment. Does Digitalk expect people to ask for Smalltalk-vee or Smalltalk-five? (Some programmers have taken to

calling it ess-tee-vee.) T_EX shouldn't be spelled like tecks if it's to be pronounced teck. Vision is pronounced vih-zhun, not vih-zee-on, and any attempt to get masses of people to pronounce it wrong will meet with resistance.

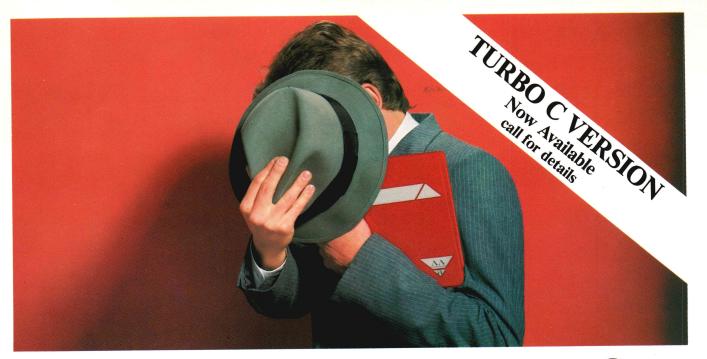
"By all means use your own name, especially if it also happens to have other, favorable connotations, as Norton suggests the authoritative Norton Anthologies. Consider changing your name to Webster or Roget. Or Knuth."

And this is from Ray Duncan:

_i_hael s_aine editor dr. do__.s_ournal _i_e _ero one _al_eston dr. red_ood_it_._a nine_our_ero si_ three dear_i_hael. _our .s_aine.s_la_es. _olu_n in the _ul_ one nine ei_ht se_en dd_ _as _uite thou_ht. _ro_o_in_. the _on_e_t o_ an en_r__tion_ethod that _an.t _e used su__ess_ull__ithout _oth a _a_hine .to en_r__t, and a hu_an .to de_r__t. _as _uite ne_ to _e. it o__urs to _e that _erha_s this rather si__le t__e o_ en_r__tion_an _e_o_e the _asis o_ a _o_er_ul _et eas_ to e__lain .to la __en. test _or .understandin_. o_ natural lan_ua_e __ _a_hines. sin_erel__ours. ra_dun_an

Wichard Swams

Michael Swaine editor-in-chief



Some Very Impressive People Keep Our Asynch C Tools Under Their Hats

This is the only way we can get some of our customers to take their hats off to us in public. We understand. Most people like to keep a good thing to themselves.

Once you see how powerful and simple our functions are, you'll want to treat our Communications Library Plus as a well guarded trade secret too.

Essential provides the best alternative to Assembly

language for communicating via an RS-232.

However, should curiosity get the best of you, call us at 201-762-6965 and we'll let you in on the identity of some of our secret admirers.



great program is a great library

What Good Are Library Functions If You Can't Get Them To Work?

Providing functions is not enough. Essential Communications Library Plus includes BreakOut, a slick on-line data monitor. BreakOut saves hours of frustration. We know, we used it to debug the XMODEM and other functions in Essential Communications.

No Royalties, 30-Day Guarantee

If within 30 days you don't find our library or BreakOut totally satisfactory, hang the whole thing up and receive a complete refund.

Functions At A Glance

- Interrupt driven to 9600 baud
- Hayes compatible support
- 150 page manual—tutorialXMODEM, XON/XOFF
- XMODEM, XON/XOFI support
- Timer/Keyboard functions
- Input buffers to 500K
- Source included
- Demo terminal program
- Demo BBS system
- All major compilers supported
- Ctrl-Break status

Comm Library Plus/BreakOut \$250 Comm Library \$185

BreakOut On-line Monitor/Debugger

- Monitor RS232 ports up to 9600 baud
- Control comm variables and line signals
- Send/receive data using the scratch-pad editor
- Edit scratch-pad in Hex or ASCII
- Save data to file or re-transmit it
- User configurable keyboard macros
- Use symbols for control chars (<ACK>) for Hex 06

BreakOut \$125

Do Your Homework

The library you buy will influence the rest of your programming life. When you've done your homework, you'll choose Essential. Call our support staff of C pro-

grammers and find out now how things will be after your check clears.

To order or for support call: 201-762-6965

For foreign orders contact: England: Gray Matter Tel. (0364) 53499

Japan: Lifeboat Inc. of Japan Tel: 293 4711 West Germany: Omnitex Tel. 07623-61820

Essential Software, Inc.

P.O. Box 1003, Maplewood, New Jersey 07040

